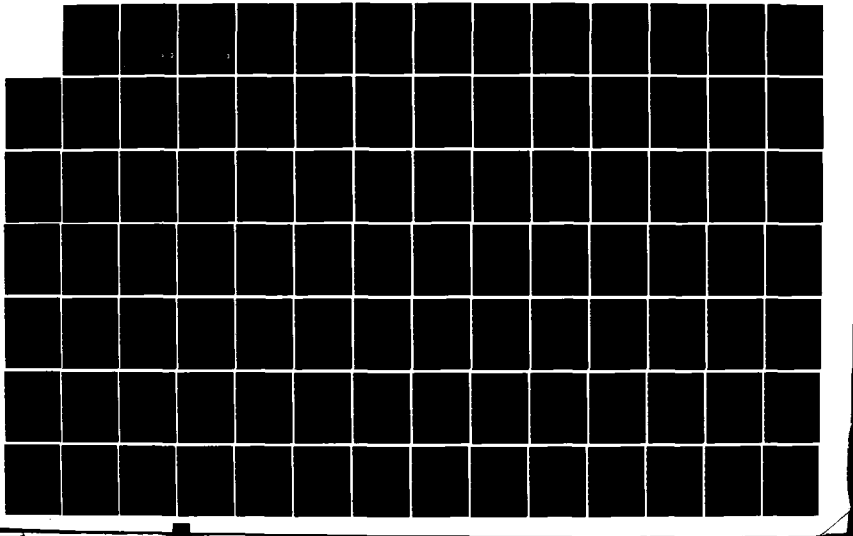
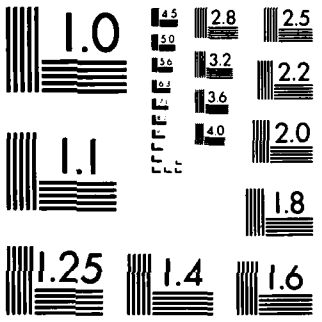
AD-A151 706 SIMULATION MODEL OF A CSMA/CD BUS LOCAL AREA NETWORK WITH MULTIPLE VARIABLES(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J M SCHRIML 1/3

UNCLASSIFIED DEC 84 AFIT/GE/ENG/84D-57 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(1)

SIMULATION MODEL OF A CSMA/CD BUS LOCAL AREA NETWORK WITH MULTIPLE VARIABLES

THESIS

John M. Schriml, B.E.E.
Captain, USAF

AFIT/GE/ENG/84D-57

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

SIMULATION MODEL OF A CSMA/CD BUS LOCAL AREA NETWORK WITH MULTIPLE VARIABLES

THESIS

John M. Schriml, B.E.E.
Captain, USAF

AFIT/GE/ENG/84D-57

AFIT/GE/ENG/84D-57

SIMULATION MODEL OF A CSMA/CD BUS LOCAL AREA NETWORK WITH MULTIPLE VARIABLES

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

John M. Schriml, B.E.E.

Captain, USAF

December 1984

PREFACE

The computer simulation model developed by this report will allow you to determine the expected performance of a CSMA/CD bus local area network under various loads. This simulation model is the first model that I have seen which will allow you to load up to 500 input sources, all with a different input rate and packet length. I feel this simulation model is one of the most realistic simulation models developed to date, because of the multiple variables of input sources.

I wish to express my thanks to Major Walter D. Seward, my advisor, and Captain David A. King for their assistant and patience during our technical discussions. In addition, I would like to extend my gratitude to Anne Slone of HQ AFLC, my sponsor, and her associate, Larry Johnson, for their support of time and the use of their micro-computer. A special thanks goes to my proof reader of 35 years, my mom, and finally, a special appreciation to my wife, Rieko, my daughters, Mana and Michele, and my soon to be son, Joseph, for their support, love and patience during the time they had to put up with a part-time husband and father.

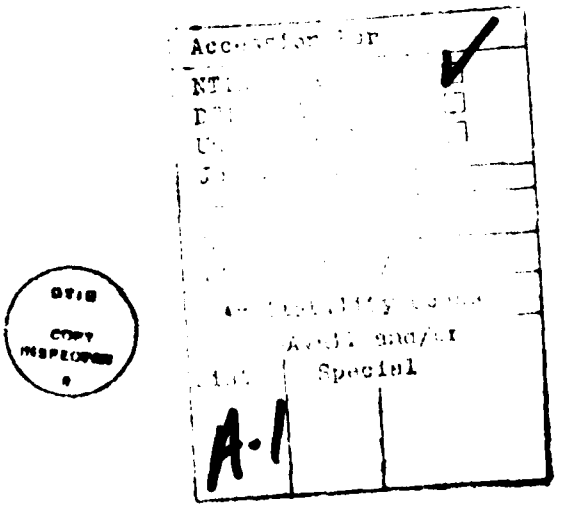


Table of Contents

List of Figures

List of Tables

ABSTRACT

Numerous statistical studies of the expected performance of a CSMA/CD bus local area network have been completed. With the statistical approach, the number of input sources is normally limited. Also, input sources normally possess the same statistical parameters with respect to input rate and the amount of data per input. The CSMA/CD bus local area network simulation model developed by this thesis can handle up to 500 input sources, all with a different input rate and amount of data per input. The limitation of 500 is due only to the fact that the simulation model was implemented on a 128K memory micro-computer. The number of sources can be increased by implementing the simulation model with a computer with a larger memory.

The simulation model takes the approach that once an input is made, the time for the input to travel through the various stages of a network can be easily calculated. Therefore, the simulation model generates traffic based on the statistical parameters of each individual source, then tracks the input as the simulation clock ticks. Using the memory power of the computer to keep track of the location of all inputs, the simulation model is able to determine the effect of an input on all other inputs. In some cases, an input has no direct effect on other inputs, and at the other extreme, when inputs want to use the bus at the same time, they have a drastic effect on each others performance. Numerous tests were performed to demonstrate the ability of the simulation model to model a CSMA/CD bus LAN. The simulation model will accept the following multiple variables prior to each simulation run: data rate of the bus, length of the bus cable, overhead bits of the bus, actual delay

of the bus cable, data rate of each source terminal, time interval between each input, amount of data per each input, and whether the CSMA/CD is 1-persistant or p-persistant.

I. INTRODUCTION

This introductory chapter begins by providing a background as to why a bus local area network (LAN) simulation model is needed. In addition, this chapter defines the scope of the simulation model that is developed by this thesis. Also, it provides the author's interpretation of the current knowledge of network models and the approach this thesis will take. Finally, it concludes with a brief overview of the remaining chapters and appendixes of this thesis.

Background

Air Force Logistic Command (AFLC) has decided to procure commercially available data communication equipment to interconnect asynchronous devices over a broadband local area network (LAN). The LAN is to be installed in buildings 262 and 266, HQ AFLC, at Wright-Patterson Air Force Base, Ohio. The bus cable topology being employed will have over 9000 outlets for computer and terminal connections. The bus cable being used has a frequency operating range of 40 MHz to 400 MHz. The cable's frequency spectrum can be divided into 60 channels by using broadband equipment, each channel with a 6 MHz bandwidth. Since each channel operates at a different frequency, the terminal/computer bus interface unit (BIU) must operate at a given channel's assigned frequency. How many BIUs for a given channel should be bought/installed? In order to answer this question accurately, it must first be determined how many sessions (established communication links) can be supported on one channel, given a maximum amount of allowable delay. The number of sessions vs required channel performance would be determined by trial and

implemented to delay how soon a BIU can attempt a re-transmission after a collision. One method used to determine how long a BIU must wait before attempting a re-transmission is called binary exponential backoff. Tanenbaum (7:294) describes a binary exponential backoff in this manner. "After a successful transmission, all BIUs may compete for the first contention slot. If there is a collision, all colliding BIUs set a local parameter, L, to 2 and choose one of the next L slots for re-transmission. Every time a BIU is involved in a collision, it doubles its value of L. In effect, after k collisions, a fraction 2exp(-k) of the BIUs will attempt to re-transmit in each of the succeeding slots. As the LAN becomes more and more heavily loaded, the BIUs automatically adapt to the load." The binary exponential backoff used by this thesis, uses a modified version of the one described by Tanenbaum. The thesis defines a slot as the amount of time that the BIU wants to use the busy. After the first collision, the BIU randomly picks some waiting time between 0 and 2 x the time interval the BIU wants to use the bus. If another collision occurs, the BIU randomly picks some waiting time between 0 and 4 x the time interval the BIU wants to use the bus. As with Tanenbaum version, as the LAN becomes more and more heavily loaded, the BIUs automatically adapt to the load.

Summary

Local area networks are becoming a popular and efficient means to transfer local information. Within the next few years more and more military organizations will be utilizing LANs. From the current trends, it can be expected that the majority of these new military LANs will be bus topology, using the broadband and CSMA/CD configurations.

bus is found busy. Persistent can be defined into three classes; 1-persistent, p-persistent, and non-persistent (7:289-291). When a BIU has data to send, it first listens to the channel to see if the bus is busy. If the bus is busy, the BIU waits until it becomes idle. When the BIU detects an idle channel, it transmits its data. This type of protocol is called 1-persistent because the BIU transmits with a probability of 1 whenever if finds the bus idle. With the second class, non-persistent, an attempt is made to be less greedy. If the bus is detected to be in use, the BIU does not continually sense it for the purpose of seizing it immediately upon detecting the end of the previous transmission. Instead, it waits a random period of time and then repeats the algorithm. The third class, p-persitent, falls in between the 1-persistent and non-persistent protocols. Like the 1-persistent, the BIU for the p-persistent senses a busy channel for the time when the bus goes idle. But when the bus does go idle, there is only a probability of p that the BIU will transmit, with a probability of $q = 1 - p$ it will defer until later to transmit. The p-persistent is more greedy than non-persistent but not as greedy as persistent. The p-persistent and non-persistent leads to better channel utilization and longer delays than 1-persistent.

Binary Exponential Backoff. Even though a CSMA/CD senses the bus cable to avoid collisions, collisions do occur when two or more BIUs begin transmitting at approximately the same time. Once the BIUs detect a collision they stop transmitting. If these same BIUs tried to transmit again as soon as the collision clears up, another collision will occur. To break this cycle of repeated collisions, protocols have been

Token. The token management method places a token on the bus when the cable is idle. If a terminal needs to transmit, it removes the token, transmits its traffic, and when completed, places the token back onto the bus. A lack of a token on the bus indicates that the bus is busy. This method is more efficient than the slot time when there is one terminal with a large amount of traffic and others have none. As with the slot time, if the terminals have no traffic to send the channel remains idle. A problem with the token is that it does not allow equal access to all users. A terminal does not share the bus until it finishes with transmitting all of its traffic.

Contention. The contention method is called random access. That is, when a terminal has information to transmit and the bus is idle, it transmits. All terminals are in contention for the cable media. The most popular version of the contention method is the Carrier Sense Multiple Access/Collision Detection (CSMA/CD) (7:291). A widely used configuration, the Ethernet, employes the CSMA/CD method (11:90). The CSMA/CD applies various rules to the randomness of the contention process in order to increase efficiency. The CSMA/CD requires that all terminals listen to the cable. If the cable is idle the terminal can transmit, and if the cable is busy, the terminal must wait. HQ AFLC has picked CSMA/CD as its LAN management system. With proper application, the contention method can be very efficient.

Persistent. The protocol used with the CSMA/CD is called persistent. This protocol is used to determine what action traffic will take if the

expensive, because of additional equipment, it is becoming the preferred method. As stated by the Sytek Corporation, "the broadband versus baseband issue is dead. Users are now sophisticated enough to see the requirements for a mix of services, and therefore media, usually on a broadband cable backbone" (9:55.5). The system being installed at HQ AFLC is a broadband LAN.

LAN Management Method

Even though a user utilizes the bus cable a very short time with respect to its input rate, the possibility of collisions occurring increases as the number of users are increased. A problem with the bus topology is that all terminals have the same path for transferring information. There must be some method to manage the terminals' access to the cable in order to prevent traffic from colliding. When using a bus topology, there are three basic methods to manage the network according to Byers (10:68): time slot, token, and contention.

Time Slot. The time slot method of management divides access time to the bus into slot intervals. Each terminal is then allocated one of these slot intervals to transmit. There are no collisions, since each terminal has a specific time interval to transmit. This method is very good in giving all terminals equal access to the cable. It is also a very efficient method when all terminals have data to transmit. A problem with the time slot method occurs when one user has a large amount of traffic to transmit and the others have little or no traffic to send. This results in the channel being idle (wasted) for most of the time, even though one terminal has data to send.

Bus Interface Unit

The bus interface unit (BIU), as indicated earlier in Figure 2, is the device which connects the user to the bus cable. On the bus side, the BIU can transmit and receive data in the megabit/sec range. On the user side, the BIU normally accepts and sends data at a rate between 75 - 19,200 bits/sec. The BIU implements the protocol for the network, such as, initialization, framing, link management, error control, flow control, transparency, abnormal condition recovery, and user data encryption. As stated by Hopkins (8:232), "the BIU accepts data from the subscriber, buffers the data until the channel is free, and then transmits the data as an addressed packet. The BIU also scans each packet on the bus for its own address. If the packet is intended for the subscriber, the BIU reads the complete packet from the channel into a buffer, and then clocks the data to the subscriber device at the proper data rate."

Baseband vs Broadband

There are two basic methods to utilize a bus cable topology, baseband or broadband (6:157). The baseband method allows the terminals access to the entire frequency spectrum of the cable. In effect, this limits the cable to one channel. The other method, broadband, allows the terminals access to only a portion of the cable's frequency sprectrum. The unused portion of the cable's frequency spectrum can be used for additional channels or other uses. The broadband system is very similar to the home cable television system except that in addition to receiving, there is also a capability to transmit. Even though the broadband system is more

The bus is simply a cable that is routed throughout the organization's office areas. Terminals are then tapped into the bus cable. The bus gives each user the ability to interact directly with all other users and host computers. A simple example is shown in Figure 2 to help illustrate why numerous users are able to utilize the same bus cable. Figure 2 depicts a user inputing 1 character of information every 200 msec (milliseconds).

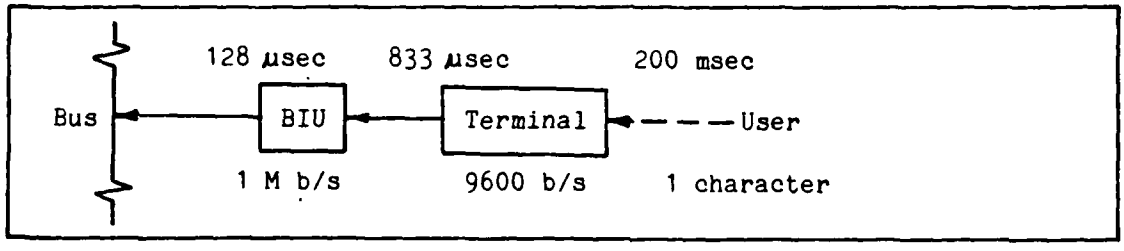


Figure 2. Example of 1 Character Input to Bus.

The terminal then transposes the 1 character of information into an 8 bit ASCII code plus parity. The 8 bit code is then output at a rate of 9600 bits/sec. It will take approximately 833 µsec (microseconds) for the modem to transfer 8 bits of information to the bus interface unit (BIU). It will then take the BIU, operating at 1 megabit/sec, approximately 128 µsec to transfer the information (plus 120 bits of overhead) along the bus. Based on the user's input rate and the bus's operating speed, the bus is idle 99.936% of the time. Even if the input rate was increased to 1200 characters/sec, the maximum rate the terminal could handle (9600 b/s), the bus would still be idle 99.028% of the time. The quick response of the bus, in comparison to rate at which a user inputs data, allows the same bus to be shared by numerous users.

planning and acqusition stages of putting in LANs at its Headquarters and all five of its Air Logistic Centers. (Long term plans will interconnect these centers with the HO). By the middle of 1985, HQ AFLC's LAN should be operational. In the near future, government personnel will be expected to utilize, engineer, monitor, and/or maintain local area networks as part of their routine duties.

LAN Utilization

The usefulness of a local area network is limited only by the user's imagination. In the case of HQ AFLC, the prime use will be to assist in the management of the Air Force's logistic resources. With the Air Force's resources distributed all over the world, the LAN will provide virtual real time location and accountability of these resources. Possible secondary uses of the AFLC network may be for electronic mail, security monitoring, live video broadcasts, and fire detection.

Bus Topology

One of the most utilized topolgies for a LAN is the bus. The bus topology has been choosen by HQ AFLC for its network. A bus topology is illustrated in Figure 1.

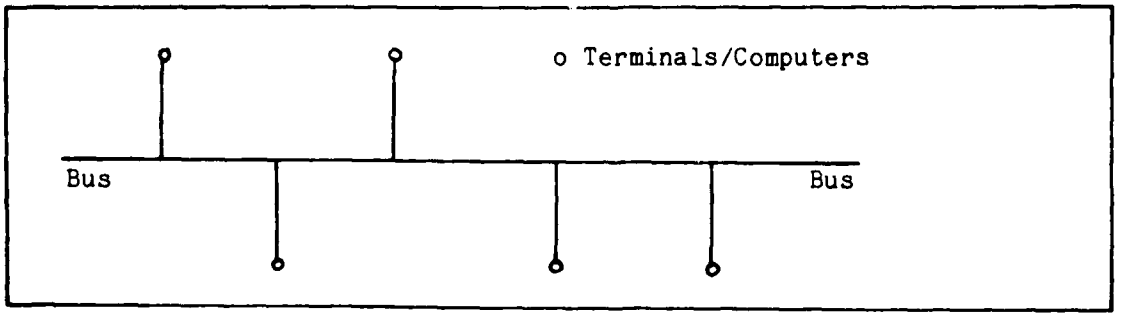


Figure 1. Bus Type Local Area Network.

II. THEORY-LOCAL AREA NETWORKS

Introduction

The local area network (LAN), with the advent of the information explosion, is the natural outgrowth of the need, and finally, the ability (due to the micro-computer) to exchange information between locations. This chapter briefly discusses some of the theory behind a bus local area network.

Definition

"A local area network is a network within a small confined area and it is the piece of the communication net that interconnects equipment such as word processors, computers, terminals, and telefacsimile machines and ties them altogether" (6:137). Tanenbaum (7:286) describes the local area network as generally having three distinctive characteristics:

- A diameter of not more than a few kilometers
- A total data rate exceeding 1 Mbps
- Ownership by a single organization

LAN's Importance

The importance of local area networks, with respect to the military, should be quite evident. The difference between success or failure of a military operation can depend heavily on the ability to gather and exchange information. In the past, LANs were normally confined to the academic environment. But within the next few years a rapid increase in the military's use of the LAN is expected. Already HQ AFLC is in the

approaches in designing a simulation model. Chapter 4 provides the design concepts, model assumptions and implementation, and the simulation algorithm for the bus simulation model developed by this thesis. Chapter 5 provides the test results of various simulation runs made with the thesis model and Chapter 6 discusses some recommendations. Appendix A provides the User's Guide and examples of a simulation run. Appendix B provides the actual Pascal computer programs, and finally, appendix C provides the design input limitations of the network model.

- Adjust for the various numbers of characters per input
- Adjust for the terminal/modem to BIU bit rate
- Adjust for the bus bit rate
- Adjust for overhead bits
- Adjust for cable length and added delay
- Generate various traffic loads
- Monitor all traffic
- Collect data on:
  - -- Number of inputs
  - -- Number of collisions
  - -- Delay due to collisions
  - -- Delay due to the bus being busy
  - -- Delay due to flow control
  - -- Amount of data being inputed
  - -- Traffic being passed by the bus
- Set the amount of simulation time
- Clock and keep track of all events
- Store necessary data for analysis
- Analyze simulation results

The computer programs are flexible enough to allow for hundreds of customers and for each customer to be transmitting at a different data rate.

Overview of Remaining Sections

Chapter 2 provides the novice with some of the theory behind a bus local area network. Chapter 3 provides some insight on the various

to amount of traffic a station can input. With these assumptions, many of the input variables reduce to statistical parameters. In addition, results of these previous models normally provided a chart depicting network performance as delay vs inputed traffic. The results of this thesis model indicate that a simple delay vs inputed traffic chart does not adequately define the performance of a network. The thesis model demonstrates that sessions which input different amounts of traffic, will have different amounts of delay. Network models built with statistical parameters are very good for comparing the performance of the various types of networks against each other, but they are very poor for analyzing an individual network where all input parameters could be different.

Approach

As indicated above, previous network models have been developed primarily for academic discussions. The anticipated solution to modeling the AFLC network is to build a scale model. In this case, a scale model does not imply a model that looks like the AFLC network, but a model that exhibits all the time characteristics of the AFLC network. Having researched the various characteristics of typical local area networks, it was necessary to construct a computer program that modeled the time characteristics of an actual LAN. The requirements for the model developed in this thesis include computer programs that are able to:

- Select typical sessions from a file
- Adjust for the various inputs rates of the sessions
- Adjust for the deviation of the various input rates

are the work of Tokoro and Tamaru for Acknowledging Ethernet (2), Franta and Bilodeau for Priority CSMA, Hughes and Li (3) for Ethernet, and Almes and Lazowska (4) for What are Termed Ethernet-Like Networks. These studies are all directed at low level protocols and few consider more than a dozen or so stations. Apparently neither the experimental nor simulation approach has adequately addressed the performance of CSMA/CD networks loaded with hundreds of stations and including several protocol layers."

Even though O'Reilly and Hammond (1) developed a model which takes into account hundreds of station, their model still limits itself to the fact that all stations have one mean packet length and all stations have identical Poisson arrival rates. In addition, as do many previous models, O'Reilly and Hammond's model disregards the effects of flow control. Their model, as does the model developed by Tobagi and Hunt (5), assumes that there are no restrictions on the amount of traffic a station can input. In reality, there are normally two restrictions on the amount of traffic a user can input. The first restriction is that the user is limited by the terminal/modem interfacing the user with the BIU, normally this rate is 9600 bits/sec. The second restriction is that most BIUs will have some type of connection, such as a RS-232 interconnection, which will limit the amount of traffic inputed if the BIU is having trouble moving the traffic it already has. Many previous models assume that all customers transmit data at the same statistical data rate, all traffic placed on the bus has the same mean packet length, and there is no limit

- The number of active sessions
- The input rate of each session
- The input rate variance of each session
- Each terminal/modem bit rate to the BIU
- The number of characters per input
- Bit rate of the BIUs
- Bus propagation delay
- Delay introduced by the active and passive components of the cable
- The CSMA/CD network media access scheme
- The BIU overhead

Originally, the model was planned to be limited to the performance of a single channel. Traffic generated by any future gateway connections, to other AFLC LANs, would not be considered. Actually, if the average amount of traffic from gateways and other channels can be anticipated and represented as sessions, this model can be used.

Summary of Current Knowledge

Numerous articles have been published in recent years concerning the performance of various local area networks. These articles have discussed mathematical, statistical, and computer simulation models of various networks. The main problem with these articles has been that they dealt with the ideal case. As stated by O'Reilly and Hammond (1:427),

"A number of simulation studies have been reported in the last several years. Typical of these studies

error unless a formal method is developed to predict LAN performance. Such a trial and error method would result in wasted effort, in time and money.

Problem

The problem as stated by the thesis proposal of HQ AFLC/LMSC is, "we need a model of the proposed network to determine what performance we can expect under various loads and to help make decisions such as how many users can be accommodated on a channel. The model is to aid us in effective network management." In short, the problem is that without some indication of a channel's performance, the channel will be loaded with customers until finally it reaches an unacceptable level of performance. Or a slightly different viewpoint, a channel may be operating satisfactorily and there may be a new requirement to add more customers. Will the channel support these new customers or should a new channel be added? If an attempt is made to add more customers to a channel and its performance drops below an acceptable level, resources in time and money would have been wasted. In addition, if a new channel is added when it is not necessary, again resources would have been wasted. The problem of inefficient network management can be overcome by designing an accurate model of the network with delay and channel utilization as output parameters.

Scope

The local area network simulation model developed by this thesis takes into account the following parameters:

III. SIMULATION MODELS

Introduction

This chapter touches on some of the methods used to build simulation models. The three areas dicussed in this chapter are; the resident vs transient viewpoint, continuous vs discrete event, and using minicomputers and Pascal for model building.

RESIDENT vs. TRANSIENT

In Lee Schruben's article, "Modeling Systems Using Discrete Event Simulation"(12:101-102), he discusses two categories or viewpoints in modeling systems using simulation, resident and transient. He illustrates the two categories by comparing the two approaches in simulating a factory. A simulation model of a factory using the resident viewpoint would take into account the machines, workers, storage spaces, material handling equipment, production schedules, etc. The transient point of view would take into account the parts being produced by the factory. For example, he states, we might model the factory by describing what happens to parts as they travel through the production process or we might model the same factory by describing what happens to the machines and inventories.

The local area newtork simulation model developed by this thesis takes the transient point of view. That is, the thesis model simply monitors all traffic as it enters, travels through a theoretical network, and finally as it exits the bus.

Continuous vs Discrete Event Simulation

A continuous simulation involves a simulation whose clock counts through each increment of time. A continuous simulation model would be used if the states within the model change often with respect to the simulation increment of time. In contrast, discrete simulation involves the realization of a system model that changes state only at discrete points in time, called events (13:41). In developing the thesis model, discrete event simulation was used. In the thesis model, instead of counting each increment of time, the model determines when the next important event will occur, and then steps the clock to that event.

Pascal and Micro-computers

As stated by Seila and Chan (13:41), in order to realize a discrete event simulation, the language used must provide facilities for representing entities, attributes and sets, for manipulating the entities in sets (inserting and removing them, and searching through the set), and for doing scientific computations. The simulation languages SIMSCRIPT, SIMULA, GPSS, GASP, and SLAM, which are normally available only on large mainframe computers, provide these features.

The author of this thesis being unfamiliar with these simulation languages, selected the Pascal language to write the simulation programs. (Pascal was selected because the author had an opportunity to study Pascal prior to the start of the thesis project.) To support the author's selection of Pascal, Seila and Chen article (13:42) further states, "that their research showed that it is possible to develop discrete event simulation using Pascal with essentially the same level of

effort that is required using a special purpose simulation language. This means that relatively sophisticated, realistic simulations are not only possible, but practical on microcomputers costing as little as $3,000 - $4,000. The limitation on the possible size and complexity of the simulation is imposed primarily by the size of the computer memory and the programmer's imagination."

Summary

The simulation model developed by this thesis uses the transient point of view, discrete events, Pascal programming language, and a micro-computer.

IV. BUS NETWORK DESIGN AND MODELING

Introduction

This chapter reviews the design concepts of the bus simulation model created by this thesis. It explains the difference between sessions and customers, and provides the assumptions made with the simulation model. Finally it concludes with discussions of how the model was implemented and how the simulation algorithm works.

DESIGN CONCEPT

The design concept of the network simulation model is very simple. It assumes that, given the type of traffic being input into a network, the performance of a network can be determined by monitoring the time it takes for the traffic to progress through a theoretical network. The design concept relies heavily on the memory capabilities of the computer to keep track of the location of all traffic. As stated earlier, the simulation takes the transient point of view. That is, the simulation model is only interested in monitoring the time it takes traffic to progress through a network. Also the concept relies on the fact that the time it takes for the traffic to travel through the various stages of the network can be easily calculated. A simple example of these calculated times is as follows:

EXAMPLE. Suppose it is known that one customer is inputing one character of traffic every 200 msec with an input deviation uniformly distributed between +/- 10 msec. With this information, it is known that an input, on the average, will occur every 200 msec. It is also known

that one character of information will generate 8 bits of traffic (7 bit ASCII code plus a parity bit). It can also be calculated that a terminal, capable of transmitting data to the BIU at a rate of 9600 bits/sec, will need approximately 833 usec to transfer the data to the BIU. At the BIU, an overhead of 120 bits are added to the 8 bits of information. It can then be calculated that a bus operating at 1 megabit/sec will need 128 usec to pass the information.

From the example, it can be seen that once an input is made, the times required for the input to progress through the stages of a network can easily be determined. If this idea is taken one step further and the time for the first input is randomly selected, then the time interval for the second input can be randomly picked to occur somewhere between 190 and 210 msec later. If the simulation model is provided the randomly picked time for the first input and the expected interval for the next input, and the model has already calculated the times required for the input to travel the network, then the simulation model (computer) can keep track of the location of the traffic as the simulation clock counts. The simulation model can be expanded to include as many customers as desired, with the only limitation being the memory capacity of the computer being used. These additional customers can all have different input rates and different amounts of traffic being inputed. The computer simply calculates the individual travel time and input interval for each customer, and adjusts their location in the theoretical network accordingly as the simulation clocks counts.

The design concept, as stated in the beginning, is very simple. The

simulation model simply generates traffic based on the types of customers loaded, then tracks the traffic as it would normally progress through a theoretical network. At any instant in time, the memory of the computer has the exact location of the traffic generated. Data collected on any difficulties of the traffic to pass through the network will provide the performance of the bus network being simulated. Before going into more detail on how the concept is implemented, it is important to clarify two terms, customers vs sessions.

CUSTOMERS vs SESSIONS

In order to eliminate any confusion, it is important to define the difference between customers and sessions. In a local area network, a customer may wish to communicate with another customer or a host computer. A communication link (session) must be established in order to communicate. A session, as used by this thesis, is composed of the data rate of the terminal, number of characters inputed with each input, the average amount of time between each input, and the maximum +/- deviation from the average time between inputs. The +/- deviation of the input is uniformly distributed between the maximum +/- values of the deviation. The simulation model developed by this thesis, simulates sessions, not necessarily individual customers. The model simulates a network based on the number and types of sessions a group of customers may generate. For example, a LAN may have a 1000 customers connected to it. The actual performance of the network depends directly on the type of traffic each of these customers generate. If each customer seldom used the network, the perfomance of the network could be expected to be satisfactory. On the other hand, if all customers tried to transfer a large amount of data

at the same time, network performance could be expected to be unsatisfactory. Therefore, it is impossible to determine network performance based solely on the number of customers attached. The performance of the network can be determined if the expected type of traffic (sessions) generated by the 1000 customers is known or estimated. It is acceptable to think of the sessions being simulated as customers, but it must be realized that the customers/sessions being simulated, is the amount of traffic being produced by the entire population of customers attached to the network and that a simulation customer/session is a typical customer, not necessarily a particular customer attached to the network.

MODEL ASSUMPTIONS

The following assumptions were used in designing the bus, CSMA/CD (1- and p- persistent) simulation model.

- Each BIU can sense the traffic of all other BIUs.
- The amount of traffic input by a particular session is constant.
- The amount of traffic input by various types of sessions may be different.
- The time interval between each input of a particular session may vary.
- The sessions/customers being simulated is a composite of the traffic produced by the entire population attached to the LAN.
- Flow control allows the BIU to handle only one input at a time
- The BIU, when starting to transmit data, will wait for the time interval equal to two round trip delays of the bus cable before allowing

new traffic to be inputed.

- Traffic waiting at the bus to be transmitted, will wait an additional time interval equal to one round trip delay of the bus before transmitting.

- A modified version of the binary exponential backoff is used for collisions. The local parameter used for binary exponential backoff is reset to 2 after each successful transmission.

MODEL IMPLEMENTATION

This thesis has implemented the LAN simulation model around seven Pascal computer programs. The user's guide and examples of the programs outputs can be found in Appendix A. The actual programs can be found in Appendix B and the input limitations of the network simulation model can be found in Appendix C. The names of the seven program are:

1. Input_t(ypical sessions)
2. Sessions
3. Input_n(etwork sessions)
4. Network
5. Parameters
6. Simulate
7. Evaluate

These programs and how they interact with created files and user inputs are shown in Figure 3 on the next page. A brief description and function of each program follows:

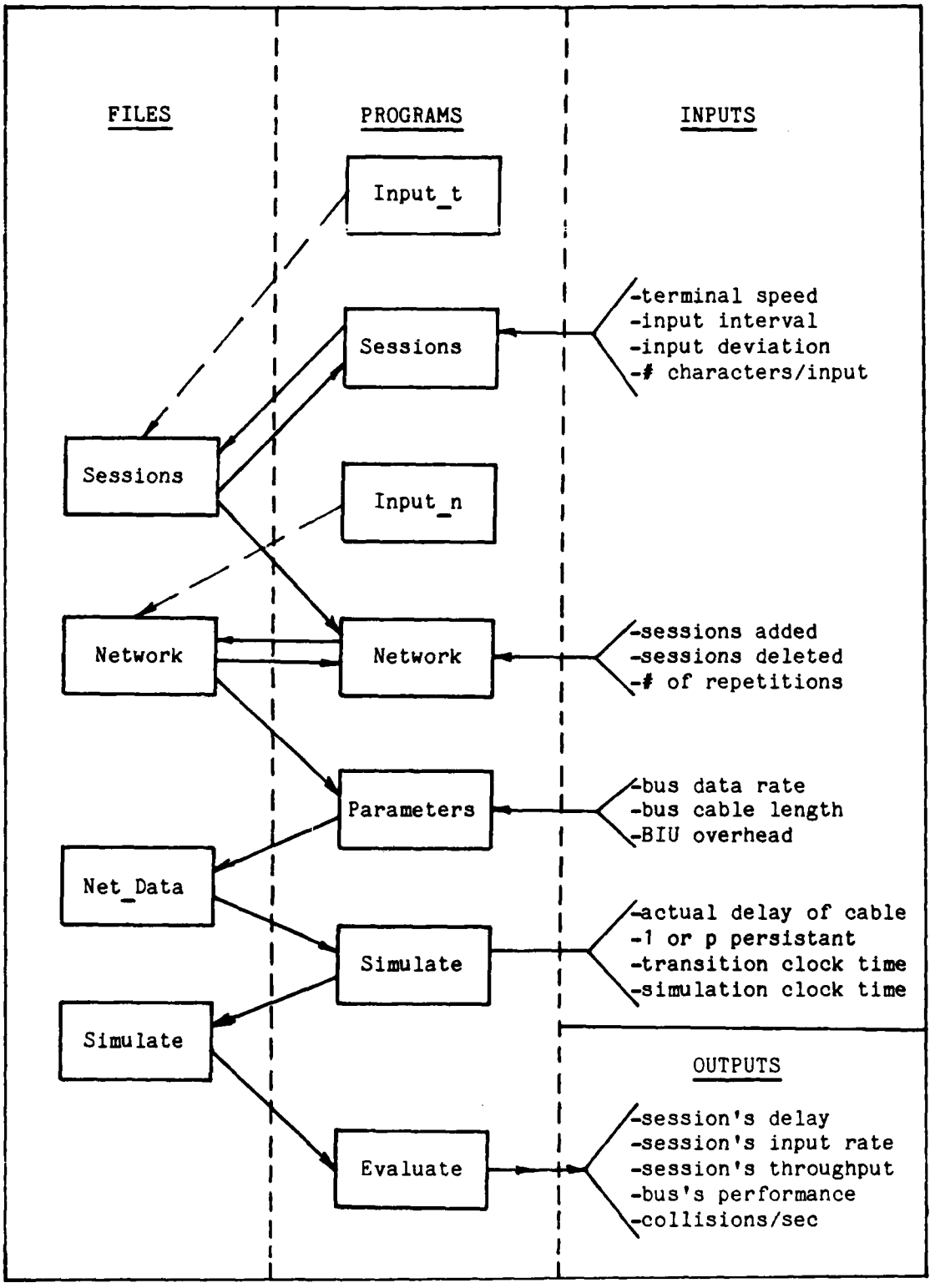


Figure 3. Model Implementation

Program Input-t(ypical sessions). The function of this program is to establish a file which will contain the list of typical customers/sessions which can be selected for simulation. The file is established after the program prompts the user for the terminal data rate, number of characters per input, time interval between inputs, and the time deviation of inputs for the first typical session. The time deivation entered, is the maximum +/- deviation uniformly distributed around the average time interval between inputs. The program then prompts the user to use 'program Sessions' for any further deletions and/or additions to the typical session file.

Program Sessions. The function of this program is to make changes to the established file of typical sessions/customers. The logic behind having two programs, Input_t and Sessions, was to prevent the user from inadvertently destroying the file of typical sessions. Program Input_t creates a one customer file, while program Sessions can only make changes to that file. Program Input_t would only be used to start a file or, if desired, to destroy the complete file of customers. Program Sessions is the main program used for maintaining the file of typical sessions. The main reasons for this program are to allow the user to keep a directory of all possible sessions needed for simulation and to elimimate the user from having to continuously key in sessions for simulation.

Input-n(etwork sessions). The function of this program is to establish a file which will contain the list of sessions to be simulated. The file is established after the program prompts the user for the terminal data rate, number of characters per input, time interval between inputs,

deviation of time interval between inputs, and the number of times this session will be duplicated in the network simulation. The program then prompts the user to use 'program Network' for any further changes to the file of network customers.

Network. The function of this program is to make changes to the file of sessions that will be simulated. It allows the user to review the list of typical sessions, upon request, and prompts the user to specify which sessions are to be simulated. Again, the logic behind having two programs, Input_n and Network, was to prevent the user from inadvertently destroying the file of network sessions. Program Input_n would be used to start a file or, if desired, to destroy the complete file of network customers. Program Network is the main program used for maintaining the file of network customers. The main reasons for this program are to allow the user to build a file of customers for simulation from the file of typical customers and to eliminate keying in of customers for each simulation run.

Program Parameters. The function of this program is to calculate the travel times required for each session to travel through the various stages of the theoretical network. Calculations are made using the information from the file of sessions to be simulated and the program's prompts for data rate of the bus, round trip length of the bus cable, and the BIU overhead. The calculated times are based on the increments of the simulation clock. Increments of the simulation clock are based on 1/2 the bus cable's round trip propagation delay. That is, one tick of the simulation clock will equate to 1/2 the propagation delay of the bus

cable. There are two main reasons for using increments of 1/2 the propagation delay of the cable. First, because the simulation is counting in discrete steps of time, the simulation will appear to be more continuous and have less error, if the increments are small compared to the time for events to happen. The second reason is that collision avoidance and detection is based on the propagation delay of the cable. The simulation model is better able to handle these situations by having the clock based on increments of delay.

Program Simulate. The function of this program is to perform the actual simulation of the bus LAN. It prompts the user for the amount of time they wish the model to run before collecting data (transition time) and for the amount of time the model should run while collecting data (simualtion time). It also prompts for the type of network management, 1-persistent or p-persistent, and if p-persistent, what percentage. In addition, it prompts the user for the actual delay of the bus cable, if known. Prior to starting the actual simulation, it loads the values calculated by the program Parameters. While the simulation is running, tables collect the amount of time the bus is passing data, the amount of time the terminals are passing data, the amount of data the customers are inputing, number of inputs, number of collisions, the amount of time traffic is delayed due to the bus being busy, the amount of extra waiting time due to collisions, and the amount of time inputs are held up because the BIU can't get rid of the traffic it has. Once the simulation has stopped, the data collected is filed away for use by the last program, Evaluate.

Program Evaluate. The main function of this program is to evaluate the data collected during the simualtion. The first part of the program displays to the user the parameters that the results are based on. The second part displays to the user the performance of each type of session loaded into the network. The performance results indicate the session's delays for collisions, flow control, and the bus being busy. It also shows the amount of traffic the terminals were passing and the amount of traffic the sessions were inputing. The third part of the program indicates the total traffic being passed by the terminals and the total amount of traffic all sessions were inputing. In addition, it shows the total traffic being passed by the bus, as indicated by the time the bus was busy, as indicated by the amount of traffic the terminals were passing, and finally as indicated by the amount of traffic being inputed by the sessions.

SIMULATION ALGORITHM

In the following discussion, using a block diagram of the simulation algorithm and a sample simulation run, the simulation algorithm is explained.

In order to perform a simulation, programs Sessions and Network would be used to establish the sessions/customers to be simulated. Assuming these programs have been used, the sessions selected for this sample simulation are stored in the Network file. They are:

ID	terminal rate	# char/ input	input interval	input deviation	quantity
5	9600 b/s	1	100 msec	10 msec	2
7	19200 b/s	80	200 msec	20 msec	2

Table 1. Sessions Stored for Simulation.

From Table 1, it can be seen that one session has an typical session ID number of 5, terminal data rate of 9600 b/s, one character per input, the interval between inputs is 100 msec, the uniform deviation of the interval bewteen inputs ranges between +/- 10 msec, and finally the session is to be be duplicated twice for the simulation run.

Having selected the sessions to be simulated, program Parameters is then called. Program Parameters prompts the user for the bus data rate, round trip length of the cable, and the BIU overhead bits. For this sample simulation run, the bus data rate is 1 mega b/s, the bus cable length is 2000 ft, and the BIU overhead bits is 120 bits. With this information and the file of network customers, the program calculates the times for the traffic to progress through the various stages of the theoretical network. Table 2 depicts these calculated values.

respect to how much traffic can be handled and the expected amount of delay, for a given situation. Once the actual bus data rate for the system to be installed at HQ AFLC is known, then the model can be run to determined what kind of performance can be expected for the anticipated load. Or in the future, if the LAN is performing satisfactory and a new requirement for additional loading is made, the model can be run with the present load plus the new additional loading, to determine the impact of the new loading on the network.

Summary

In this chapter the design concept, assumptions, and the simulation algorithm concerning the thesis network simulation model were discussed. Finally the chapter concluded with a demonstration of the model's ability to provide the user with the expected network performance for a given situation.

Simulation #1. The results of simulation #1 indicate an average delay of 480.9015 msec from the time the source BIU has the traffic to the time the first character reaches the destination BIU. Also Table 8 shows that flow control was employed on an average of 301.3 msec for each input. This implies that each user wished to input blocks of data every 1000 msec, but the LAN would only accept inputs on the average of every 1301.3 msec for the given configuration. A 300K b/s bus could not handle the combined user input rate of 360K b/s and far exceeded the maximum delay of 100 msec.

Simulation #2. The results of simulation #2 indicate an average delay of 38.1645 msec from the time the source BIU has the traffic to the time the first character reaches the destination BIU. Also Table 8 shows that flow control was employed on an average of 1.233 msec for each input. This implies that each user could only input blocks of data on an average of every 1001.233 msec. The 500K b/s bus passes the 100 msec delay requirement but falls slightly short of being able to handle the 360K b/s combined input rate of the sessions.

Simulations #3 and #4. The results indicated in Table 8 for simulations #3 and #4, show that either the 750K b/s or the 1000K b/s bus can pass the 360K b/s of input data with a delay much less than 100 msec.

The demonstration was performed to show that, given the expected type of traffic being generated by the users and the configuration of the bus, the simulation model will give the expected performance of the LAN with

Simulation	Bus Rate	Actual Traffic Passed By Bus	Desired Input Rate of Sessions	Actual Input Rate of Sessions	Average Delay Due to:				
					Flow Control (1)	Bus Busy (2)	Collisions (3)	Propagation (4)	Total (2)+(3)+(4)
	Data Rate B/S				Delay in msec				
1	300K	280.5K	360K	276K	301.3	166.4	314.5	.0015	480.9015
2	500K	365.3K	360K	359K	1.233	14.489	23.674	.0015	38.1645
3	750K	366K	360K	360K	0.0	4.304	5.182	.0015	9.4875
4	1000K	366K	360K	360K	0.0	1.888	1.616	.0015	3.5055

Table 8. Simulation Results for Delay for Various Bus Data Rates.

indicated in Table 8, the following parameters were loaded into the simulation model for each of the 50 user devices:

- 9600 b/s terminal rate
- 900 characters per input
- 1000 msec between inputs
- 100 msec +/- uniform deviation from mean input rate

The parameters loaded for the bus were:

- 2000 ft maximum bus length
- 120 bits of overhead at the BIU
- Bus rate 300K b/s for simulation #1, 500K b/s for simulation #2, 750K b/s for simulation #3, and 1000K b/s for simulation #4

The simulation model, after each simulation, provided the following information to the user:

- Bus rate of LAN
- Actual traffic passed by bus
- Desired combined input rate of all sessions
- Actual combined input rate of all sessions
- Average amount of time flow control was employed
- Average delay due to the bus being busy
- Average delay due to collisions

The results of the four simulations are shown in Table 8.

time the bus is busy, and the amount of time between inputs. Once the simulation is completed, these records are stored in the file Simulate until needed by the program Evaluate.

Demonstration

As stated previously in Chapter 1, HQ AFLC/LMSC indicated that, "we need a model of the proposed network to determine what performance we can expect under various loads and to help make decisions such as how many users can be accommodated on a channel." The following four simulation runs were performed to demonstrate how the simulation model developed by this thesis, fulfills that need. The scenario used for this demonstration, is based on one of the specifications identified in the 'AFLC Headquarters Local Area Network, Bus Interface Unit Specification (Draft)', dated 30 Nov 83. Summarizing the specification used for this demonstration, it states that the LAN shall provide sufficient transmission speed to support a population of 50 user devices, with each generating an average of one 900 character stream of data per second (small number of file transfer oriented user devices). Also, that the first character in a continuous character stream shall be delivered to its destination BIU device port within 100 milliseconds after the presentation of the last character in the stream to the source BIU device port under the given loading configuration. HQ AFLC was interested in knowning what kind of performance would be expected for various bus rate of a CSMA/CD system, and would the performance meet the required specification. Four separate simulations were run with the bus rates of 300K b/s, 500K b/s, 750K b/s, and 1000K b/s. The results of these four simulations are indicated in Table 8. For each of the simulations

would be found, as the clock passes through the place in time the collision occurred. In the case of p-persistent, the sessions waiting to transmit goes through an addition block, adjust further, as depicted in Figure 4. In this block sessions are randomly picked to remain at their current value in Table 3 or have an addition delay factor added to column 3, based on the percentage of the p-persistent. If more than one session remains at their present value in column 3, there will be a collison and new backup times will be added to column 3. The program recycles back to find the next event if one or zero sessions remain at their current value in column 3.

If it has not been apparent, the simulation algorithm never counts down to an actual event. It counts down to 1 clock tick before the event is to occur, evaluates the location of all network traffic and inputs, adjusts all traffic and inputs that would be effected by the next clock tick, adjusts the clock time if needed, then continues to count through the actual event. This action, of using the value of 1 to depict upcoming events, allows the simulation to determine that a zero in column 3 means that no traffic is present between the bus and the BIU. Also a zero in column 5 indicates that follow control is employed and no more traffic can be input for a particular session until the traffic already inputed is passed by the bus. A number 1 showing up in columns 3 and/or 5 allows the model to identify that an event will occur. In addition to simulating the network, program Simulate establishes an additional 7 columns of matrix space for each session simulated. These additional columns keep track of the number of inputs, number of collisions, amount of time spent waiting at the bus due traffic, amount of time flow control is employed, amount of extra time waiting due to collisions, amount of

in column 8 (row 2), and by one propagation delay of the cable. The 'adjust bus' procedure test the values of columns 3 and 5 of each session and calls the appropriate adjuster procedure (AA - JJ) to adjust each session the proper amount. Table 7 indicates the results of adjust bus.

3	4	5	6	7	8
time to bus	interval to bus	next input	input interval	input deviation	bus busy time
Time in Clock Ticks					
0	547	14164	65640	656	84
0	547	65307	65640	656	84
0	21880	83355	131280	13128	498
0	21880	36606	131280	13128	498
CLOCK:9981624					

Table 7. Simulation Adjusts for Bus Input.

Having performed 'adjust bus', Figure 4, indicates a recycle back to 'find next event'. The algorithm simply repeats itself until the clock time goes negative. The only situation not illustrated as depicted in Figure 4, is when there are more that one customer wanting to use the bus or another customer will arrive within one propagation delay of the time a BIU starts transmitting. In the case of one persistent, this situation will create a collision. Once it has been determined that a collision will occur, new backup times are randomly generated and added to column 3. Collisions, similiar to adjust input, does not advance the clock, but simply puts the sessions colliding at the appropriate place in time they

the input just input. Again since the next event is not a 1, procedure 'prepare for event' is performed. Table 6 depicts the prepare for event.

3	4	5	6	7	8
time to bus	interval to bus	next input	input interval	input deviation	bus busy time
Time in Clock Ticks					
0	547	14251	65640	656	84
1	547	65394	65640	656	84
0	21880	83442	131280	13128	498
0	21880	36693	131280	13128	498
CLOCK:9981711					

Table 6. Simulation Prepares for Second Event.

The procedure 'prepare for next event' while preparing, would have noted that there is only one input at the bus ready for transmission. Besides noting the number of sessions ready to use the bus, it checks to see if any traffic would arrive in a time interval equal to one round trip propagation delay of the cable. It knows if other traffic arrives in this interval of time, there will be a collision. Therefore, in preparing for the event, the procedure has determined that it is a bus event, only one customer is ready to transmit, and that the customer is in the second row. As depicted in Figure 4, the next step is to 'adjust bus'. Procedure 'adjust bus' notes that no collision will occur and the traffic can be passed on the bus. All other customers/sessions must be adjusted by for the next clock tick, the time the bus is busy as depicted

noted that the next event was an input event. Having prepared for the event as depicted in Table 4, Figure 4 indicates that for an input event it is necessary to 'adjust input'. 'Adjust input' simply places the value from column 4 into column 3, for the session indicating a 1 in column 5. At the same time, a new time for the next input to occur is generated by using the values from columns 6 and 7. Table 5 depicts the adjustments for an input event.

3	4	5	6	7	8
time to bus	interval to bus	next input	input interval	input deviation	bus busy time
Time in Clock Ticks					
0	547	14798	65640	656	84
548	547	65941	65640	656	84
0	21880	83989	131280	13128	498
0	21880	37240	131280	13128	498
CLOCK:9982258					

Table 5. Simulation Adjusts for Input.

It should be noted that an adjust for input does not move the clock. Tables 4 and 5 both indicate the same time. The simulation knows that an input will occur in one more tick of the clock and simply adjusts the time in column 3 by a factor of 1. It also randomly picks a new time for the next input between 65640 +/- 656 and adds 1. Having made the input adjustment, Figure 4, indicates the next step is to repeat 'find next event'. From Table 5, it can be seen that the next event will be in 548,

the accuracy of the simulation run. In this particular sample simulation, the results would be based on 304 inputs for session id #5 and 152 inputs for session id #7.) As indicated in Figure 4, the first step the program does is scan columns 3 and 5 in order to 'find next event' to occur. Since the simulation is based on discrete events, the procedure 'find next event' searches for when the next important event is to occur. In scanning columns 3 and 5, the program will note that the next non-zero event to happen is at 17743. Since the next event is not a 1, the program will then 'prepare for the event' by reducing all times in columns 3 and 5 by next event minus 1. If 'find next event' had indicated that the next event would occur in 1 tick of the simulation clock, 'prepare for next event' would have been by-passed. This action is depicted in Table 4.

3	4	5	6	7	8
time to bus	interval to bus	next input	input interval	input deviation	bus busy time
Time in Clock Ticks					
0	547	14798	65640	656	84
0	547	1	65640	656	84
0	21880	83989	131280	13128	498
0	21880	37240	131280	13128	498
CLOCK:9982258					

Table 4. Simulation Prepares for First Event.

While the program was scanning for the next event in Table 3, it also

first clock time loaded, is the transition time. The transition time simply runs the simulation for a period of time to allow transient situations to die out. The second clock time loaded, is called the simulation time. It is the actual clock time that data will be collected on the performance of the simulated network. With the clock times entered, the simulation begins. Table 2, without the ID numbers, is repeated in Table 3.

3	4	5	6	7	8
time to bus	interval to bus	next input	input interval	input deviation	bus busy time
Time in Clock Ticks					
0	547	32540	65640	656	84
0	547	17743	65640	656	84
0	21880	101731	131280	13128	498
0	21880	54982	131280	13128	498
CLOCK: 10000000					

Table 3. Start of the Simulation.

For this simulation, the clock has been set to 10000000 ticks. (Comment: The simulation time entered is with respect to the ticks of the simulation clock. One method for determining how long the simulation should run is to review the data loaded for simulation. Having reviewed the data, pick the session with the largest amount of time between inputs, then multiply that quantity by the number of inputs you wish to see simulated. The more inputs a session is allowed to input, the better

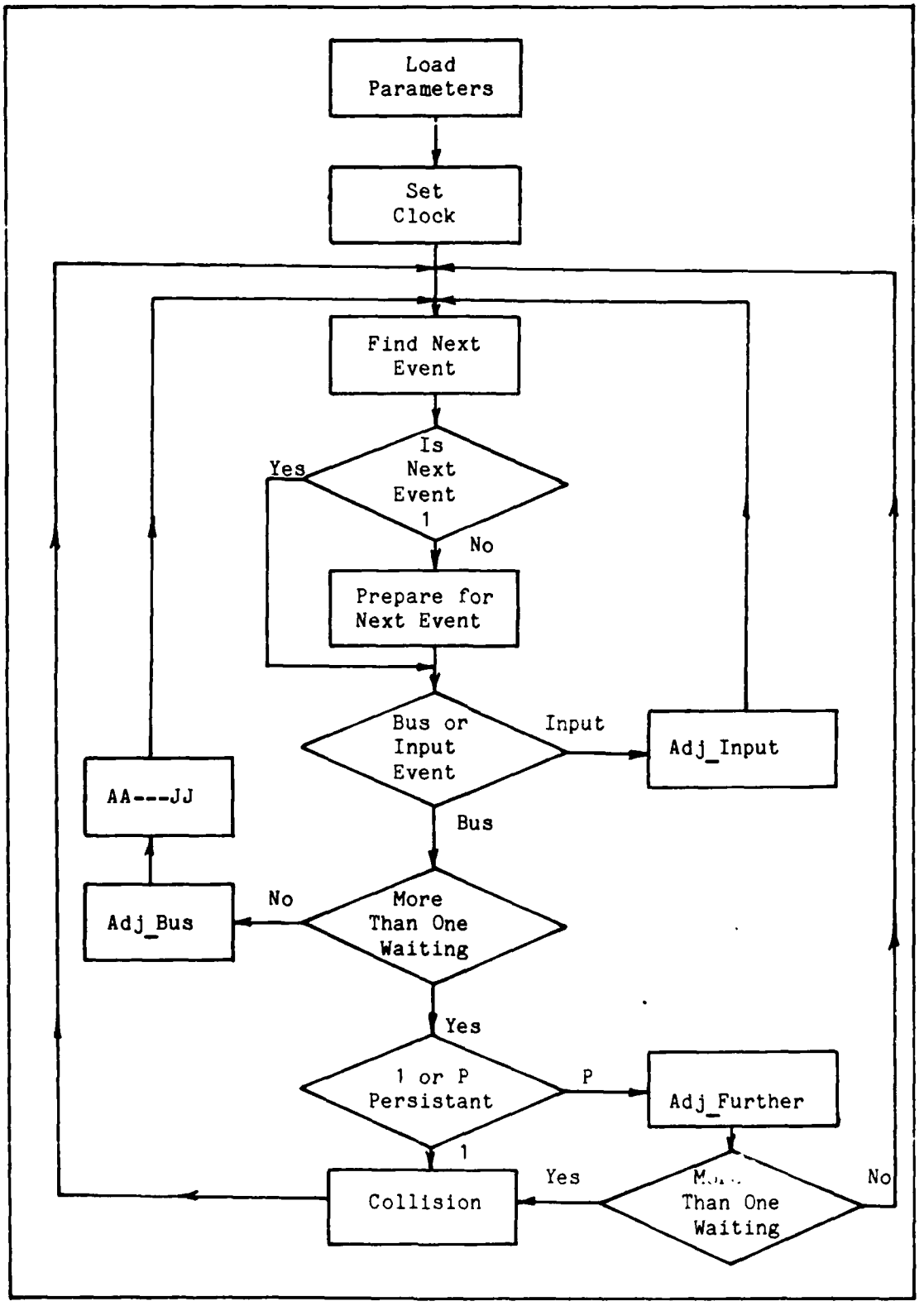


Figure 4. Block Diagram of Simulation Algorithm

terminal.

- column 4: 'Interval to bus' indicates how much time, once an input is made, that must elapse before the input reaches the bus. Once an input is made, this value is placed in column 1 and column 1 reflects the actual counting down as the input travels to the bus.

- column 5: 'Next input' indicates the amount of time that must elapse before the next input is made. This time is randomly picked by the program Parameters.

- column 6: 'Input interval' indicates the average elapse time between inputs. Once an input is made, this value, modified by the input deviation of column 7, is placed in column 3 and column 3 reflects the actual counting down of time before the next input is made.

- column 7: 'Input deviation' is the maximum +/- uniformly distributed deviation of column 6.

- column 8: 'Bus busy time' indicates the amount of time the busy is bus, once an input reaches the bus.

Once the values in Table 2 are calculated, they are stored in the file net_data. The next step is to call program Simulate. Figure 4 is the block diagram of the simulation algorithm and is used for the remaining discussion of the sample simulation run.

Program Simulate will load the calculated values produced by the program Parameters into a matrix table. The ID and extended ID are not used during the simulation and only columns 3 - 8 of Table 2 are loaded. This action is depicted in the algorithm block diagram, Figure 4. The next action of the program is to prompt the user for clock times. The

1	2	3	4	5	6	7	8
ID	ext ID	time to bus	interval to bus	next input	input interval	input dev	bus busy time
		Time in Clock Ticks					
5	1	0	547	32548	65640	656	84
5	2	0	547	17743	65640	656	84
7	1	0	21880	101731	131280	13128	498
7	2	0	21880	54982	131280	13128	498

Table 2. Calculated Values for the Simulation.

It should be remembered that the times given in table 2 are with respect to the ticks of the simulation clock, which are based on 1/2 the propagation delay of the bus cable. In this particular simulation, the interval between inputs for session #7 is 200 msec, which equates to 131280 1/2-propagation delays of the cable. An explanation of the columns in Table 2 are as follows:

- column 1: Indicates the ID number of the typical session picked for simulation.

- column 2: Indicates the extended ID number of the typical session picked for simulation. The maximum extended ID number for each type of session should match the number of times the session was to be duplicated.

- column 3: 'Time to bus' indicates the remaining time that must elapse before an input reaches the bus. A zero in this case indicates that no input has been made, and there is no traffic between the bus and the

V. TEST RESULTS

Introduction

It is impossible to test the simulation model for all the possible combinations of the variables associated with a local area network. These variables are terminal data rate, data input rate, bus data rate, bus length, bus overhead, amount of data per input, cable delay, and deviation of the input rate. The following 5 tests were performed to demonstrate that the simulation model developed in this thesis does, in fact, adequately simulate a bus CSMA/CD local area network.

Test 1

The following test was performed to validate the model's algorithm that a BIU will wait a time interval equal to two round trip delays of the bus cable before accepting more input. This waiting period of the BIU insures a successful transmission of the traffic it has, before accepting more inputs.

TEST SETUP:

-Bus

-- 1 M bit/sec data rate

-- 1-persistent

-- length of cable 2000 ft

-- overhead 120 bits

-- cable delay 3.046 μsec

-Sessions

-- number of sessions 1

-- terminal data rate 9600 bits/sec

-- input interval mean 50 msec

-- uniformly distributed input interval deviation of +/- 3 msec

-- number of characters per input 80

Synopsis: The one session being simulated is attempting to input an overall data rate of 12,800 bits/sec. It is expected that the network model will limit the session's input to approximately the data handling capability of the terminal, 9600 bits/sec.

Calculate Results: It can be calculated that the amount of time for the terminal to transfer 640 bits (80 char x 8) is 66,666.667 μsec. Also the time the BIU must wait before accepting more inputs is 6.092 μsec (2 x round trip delay). Therefore, the allowable interval between inputs that the terminal can handle is 66,672.759 μsec. This equates to the terminal passing data at a rate of 9599.1228 bits/sec.

Simulation Results: The results of the network simulation model, for a 6.134 sec simulation time, indicates a 66,680 μsec interval between inputs and traffic being passed by the terminal at a rate of 9598 bits/sec.

Test Conclusion: For the given set of parameters, the simulation model was able to provide the expected interval between inputs within .0199% and the expected amount of traffic being passed by the terminal within .0116% of the calculated values. The small amount of error is introduced by the conversion from real to integer values for the simulation.

Test Setup: Test 1 was repeated again, but this time the delay of the network was 379.341 μsec.

Calculated Results: For the given setup and the new network round trip delay, the calculated interval between inputs is 67425.349 μsec with a terminal handling capability of 9491.979 bits/sec.

Simulation Results: The results of the network simulation model indicated a 67440 μsec interval between inputs with the terminal handling data at the rate of 9489 bits/sec.

Test Conclusion: For the given set of parameters, the model was able to provide the expected interval between inputs within .02173% and the expected amount of traffic being passed by the terminal within .0313% of the calculated values. Again the small amount of error is due to real to integer conversion.

TEST 2

As discussed later in the user's guide, Appendix A, when the program Parameters is run to calculate the values needed for the simulation, a warning may be given if the calculated value for the time the bus is busy, for a given session, is in error more than 1%. An example of this is, if the bus was operating at 5 mega bits/sec, an input of one character plus 120 bits of overhead would require the bus to be busy for 16.8T, where T equals one tick of the simulation clock (T equals the time associated with 1/2 the propagation delay of the cable). Since the simulation runs on integer values, it would use the value 16 instead of 16.8. The value being used is in error more than 1% and a warning would be given. There are three methods being used to determine the amount of traffic being passed by the bus; method 1 by monitoring the time the bus is busy, method 2 by monitoring the amount of traffic being passed by the terminals, and method 3 by monitoring the amount of traffic being inputed

by the user. The warning only advises the user to disregard the results of method 1, 'traffic on the bus indicates' when using program Evaluate.

The following test was run to validate the warning and to illustrate that the other two methods can be regarded as accurate.

TEST SETUP:

-Bus

-- 1 mega bits/sec data rate

-- 1-persistant

-- length of cable 2000 ft

-- overhead 120 bits

-- cable delay 3.046 μsec

-Sessions

-- number of sessions 1

-- terminal data rate 9600 bits/sec

-- input interval 200 msec, dev 0 msec

-- number of characters per input 1

Calculated Results: With the session inputing 1 character of infomation 5 times a sec, the overall input of the session is 40 bits/sec (1 X 8 X 5). Also with an overhead of 120 bits, it can be calculated that the bus should be passing 640 bits/sec ((1 x 8) + 120) x 5).

Simulation Results: The results of the network simulation, for a 7.8 sec simulation run, indicates that the traffic being passed by the bus as indicated by method 1 is 639 bits/sec, as indicated by method 2 is 640 bits/sec, and as indicated by method 3 is 640 bits/sec.

Test Conclusion: The result using method 1 indicates an error from

the expected value of .156% In this particular case, the error produced by calculating the bus busy time is .02%. The additional error can be attributed to the integer display to the user. Methods 2 and 3 indicate no error. Also in this case, a warning would not have been given because the calculated error in the time the bus is busy is less than 1%.

TEST SETUP: Test 2 is repeated again, but this time the operating speed of the bus is 50 mega bits/sec.

Calculated Results: As indicated above, the expected traffic being passed by the bus is 640 bits/sec.

Simulation Results: The results of the network simulation, for a run of 7.8 sec, indicates that the traffic being passed by the bus as indicated by method 1 is 380 bits/sec, as indicated by method 2 is 640 bits/sec, and as indicated by method 3 is 640 bits/sec.

Test Conclusion: In this case, program Parameters gave a warning and indicated the error of the time the bus is busy was over 40% and that the results of method 1 should be disregarded. It can be seen that the error of method 1 is also in error by 40%, but again it can be noted that methods 2 and 3 provided the expected result and that a 40% error in the bus busy time used for the simulation had no real impact.

Further Discussion of Test 2

In reality, the extreme accuracy of methods 2 and 3 in test 2 can not always be expected. To help demonstrate what is happening, lets look at the way method 1 works and compare it to method 3. Method 1 simply sums up the amount of time the bus is busy because of each session. Once the simulation is completed, the total time the bus was busy is divided by the total time the simulation was run. The resulting fraction represents

the percentage of the time the bus was busy, and is multiplied by the bus operating speed. If the bus busy time being summed up during the simulation is 40% inaccurate, the resulting traffic being passed by the bus will be 40% inaccurate. On the other hand, method 3 takes a different approach.

Method 3 sums up the amount of time between each input. Knowing the amount of time between each input, can be equated to how much traffic the bus is passing. In this case, the calculated time between inputs, 200 msec, equates to 131280 ticks of the simulation clock. As the simulation runs, time between inputs is counted down by what ever event is happening next. In test 2, assuming an input is ready to be transmitted, the clock will be reduced by the amount equal to the bus busy time. Also the time before the next input, will also be reduced by the bus busy time. Because of the inaccuracy of the bus busy time (for the 50 mega b/s example), the time interval between inputs will appear to be 131279.32 instead of 131280. The 40% error in bus busy time caused an error of only .00051% for method 3, which doesn't even show up in the results.

To illustrate even further, lets take a network with a 1000 sessions all inputing 1 character at a time. Again the interval between inputs is 200 msec, but this time the bus is operating at 44 mega bits/sec. The interval between inputs is still 131280 ticks of the simulation clock. With the high bus rate and a 1000 sessions, the interval between inputs will be reduced 1000 times by the inaccuracy of the bus busy time before an input is actually made. Because of the inaccuracies, the time interval between inputs will appear to be 130380 instead of 131280. Even with this extreme case of bus busy time error of 47% and a 1000 customers, the error introduced to method 3 is only .7%.

Test 3

The following test was performed to validate the ability of the simulation model, developed by this thesis, to adequately model the performance of a CSMA/CD bus network. In Werner Bux's article [14], 'Local Area Subnetworks: A Performance Comparison', he compares the performance of various types of networks. One of the networks he compares is the CSMA/CD bus. Test 3 runs the network simulation model various times, for various loads, and compares the results with the performance curve used by Bux.

One major difference between Bux's analytical model and the thesis simulation model, is that Bux uses a continuous exponential distributed packet length, while the thesis model uses a constant packet length for each individual session. In order to validate the comparison of Bux's analytical model with the simulation model, three test simulations were run with various distributions of packet length. The results of Bux's analytical model, with bus rate of 1 M b/s and 50% loading, gave an average packet delay of 1130 μsec. The thesis simulation model using the constant packet length of Figure 5 gave an average packet delay of 1121 μsec. The simulation model using the discrete uniform distribution of Figure 6 gave an average packet delay of 1107 μsec. And finally, the simulation model using the discrete exponential distribution of Figure 7 gave an average packet delay of 1106 μsec. The factor held constant for all four packet distributions, is the mean packet length of 1000 bits. The results indicated approximately the same amount of delay (1130, 1121, 1107, and 1106 μsec) for a given mean of 1000 bits. Someone could draw the conclusion that the average packet delay is independent of the type of distribution and is dependent only on the mean packet length. This

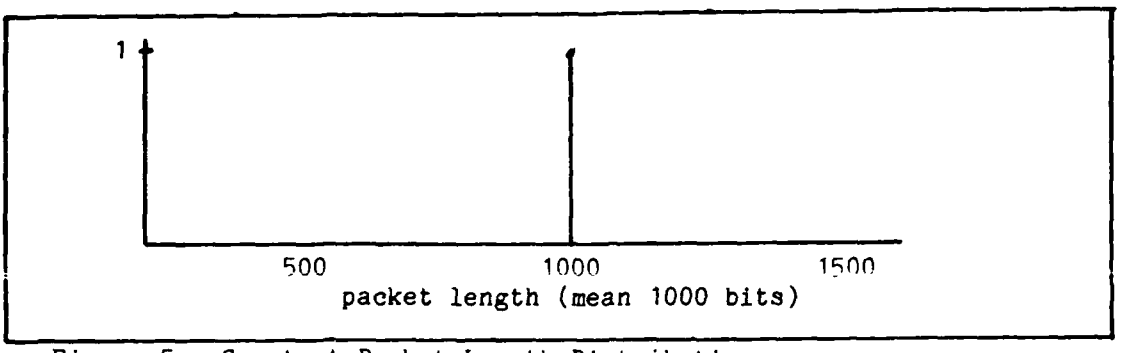


Figure 5. Constant Packet Length Distribution

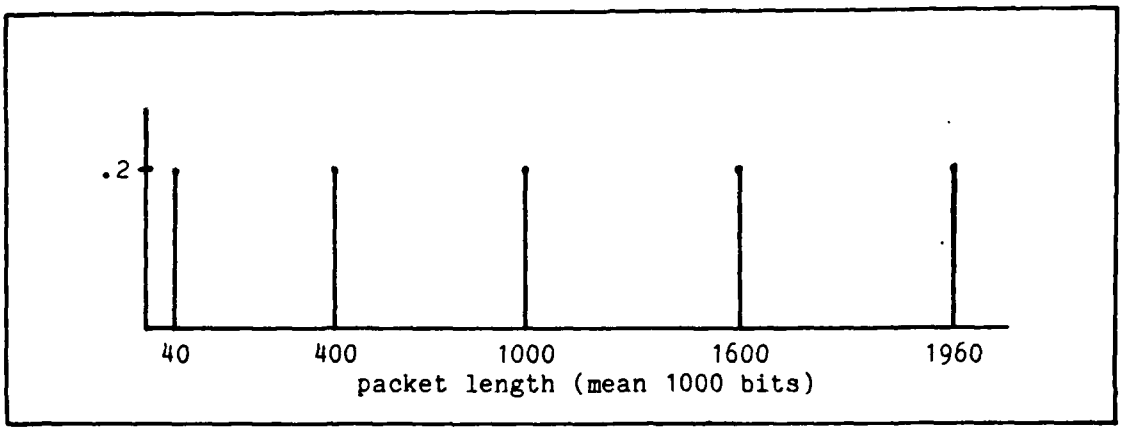


Figure 6. Discrete Uniform Distribution Packet Length.

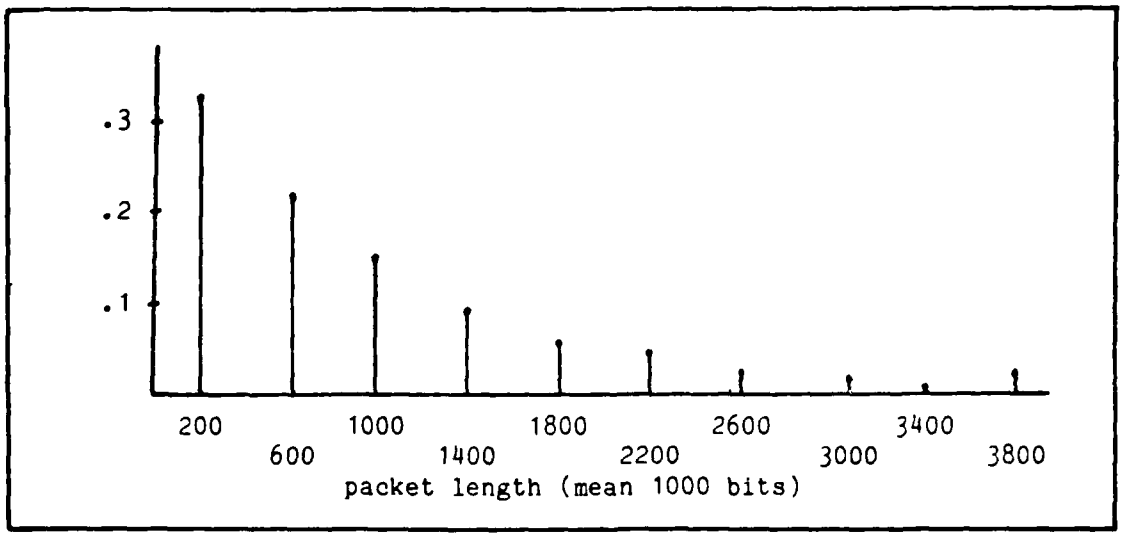


Figure 7. Discrete Exponential Distributed Packet Length.

thesis is not prepared to take this position (at this time) and would suggest further research in this area. Based on these test simulations, the thesis does take the position that it appears that it would be valid to compare the results of Bux's analytical model using exponential distribution packet length, with the simulation model using constant packet lengths. In addition, the results of test 3 indicated in Figure 8, seems to support this position. With this position in mind, the discussion of test 3 is continued.

The information provided with Bux's performance curve was:

- 1 M bit/sec transmission rate
- 2 km cable length
- 50 stations
- Exponentially distributed packet length (mean 1000 bits)
- 24 bit header
- Packets generated according to poisson process

The test setup used for the simulation model was:

TEST SETUP:

- Bus
  - -- 1 M bit/sec
  - -- 1-persistent
  - -- length of cable 6561 ft
  - -- overhead 24 bits
  - -- cable delay 9.993 μsec
- Sessions

-- number of sessions 50

-- terminal data rate 19200

-- input interval mean varied to give various loads

-- input interval uniformly distributed +/- 10% around mean

-- number of characters per input 125

Calculated Results: Bux's performance curve is indicated by a solid line in Figure 8.

Simulation Results: The results of the various runs by the simulation model developed by this thesis are indicated by the plotted points in Figure 8 on the next page.

Test Conclusion: The results indicated in Figure 8 demonstrates the ability of the simulation model to exhibit the characteristics of a CSMA/CD. To demonstate further the ability of the simulation model, an additional run was made using the packet length distribution of Figure 6. The delay results of these five groups were:

Group 1 (40 bits - 5 characters) - 505 μsec

Group 2 (400 bits - 50 characters) - 777 μsec

Group 3 (1000 bits - 125 characters) - 1066 μsec

Group 4 (1600 bits - 200 characters) - 1334 μsec

Group 5 (1960 bits - 245 characters) - 1855 μsec

The average delay vs throughput for these five groups is indicated in Figure 8 by a star. As can be seen, the curve of Bux gives the overall effect, but the simulation model developed by this thesis, can give the performance of each type of session on the network. With this particular

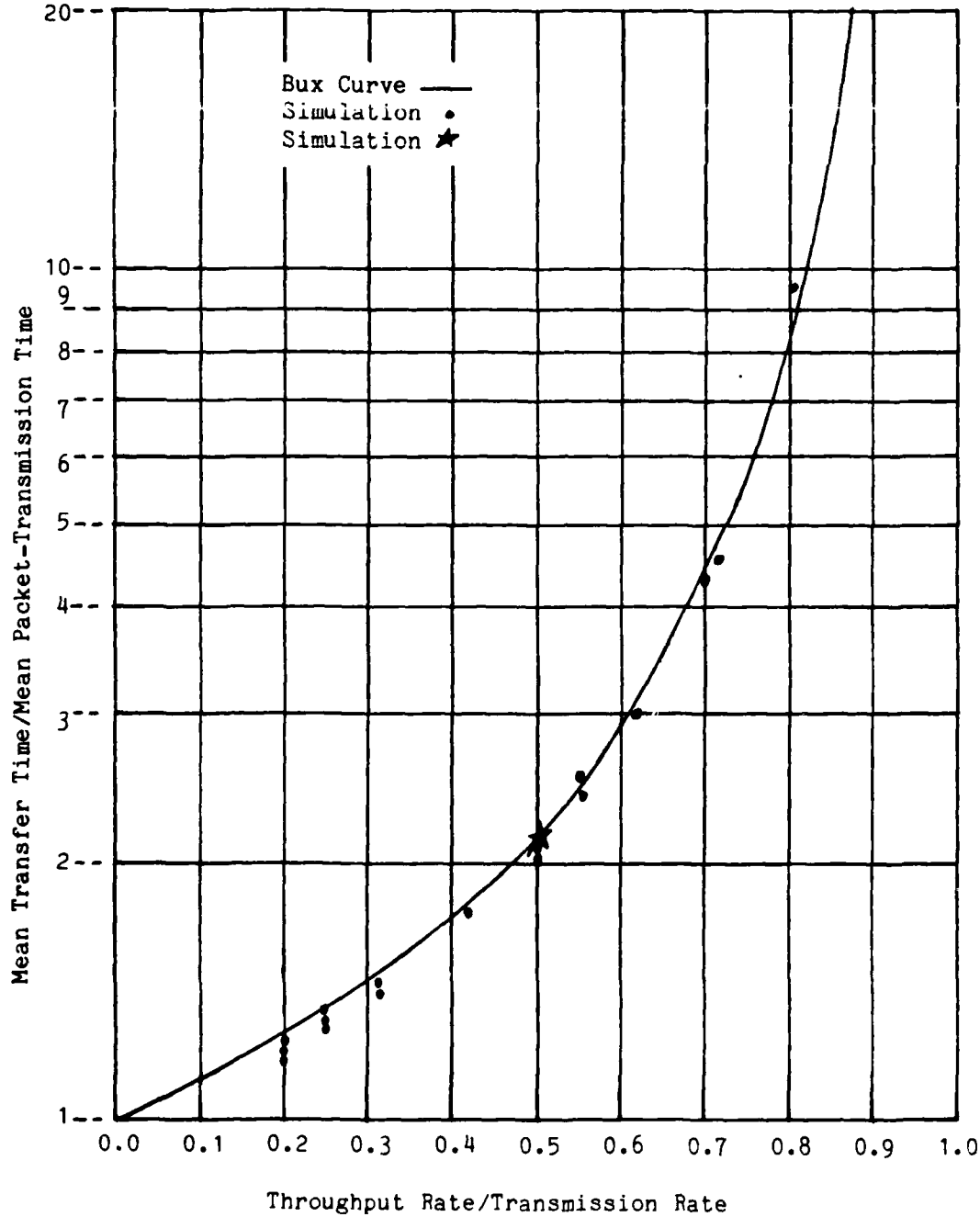


Figure 8. Transfer delay-throughput characteristics for a CSMA/CD at 1 MB/S.[14:1469]

setup, a session inputing 245 characters per input can expect a delay 3 times as much as a customer inputing data at a rate of 5 characters per input. Also in this particular case, the delay for groups 4 and 5 is great enough that they can no longer input data at the rate they desire, while the delay for groups 1, 2, and 3 is not great enough to effect their input rate. Therefore, three groups are satisfied with the network performance while the other two groups are having trouble getting their data in as fast as they wish.

Test 4

This test is basically a repeat of test 3, but the bus transmission rate has been increased to 10 mega bits/sec. The results of two simulation runs are plotted on Figure 9. The solid line is the curve used by Bux [14:1470].

Test 5

This test was performed to reinforce the users confidence in the ability of the thesis model to simulate a CSMA/CD bus network. Again the characteristic curve used for comparision came from the Bux article. In this case, he used a constant packet length. The information provided with the Bux curve was:

- 5 M bit/sec transmission rate
- 1 km cable length
- 50 stations
- packet length fixed 1000 bits
- 24 bit overhead

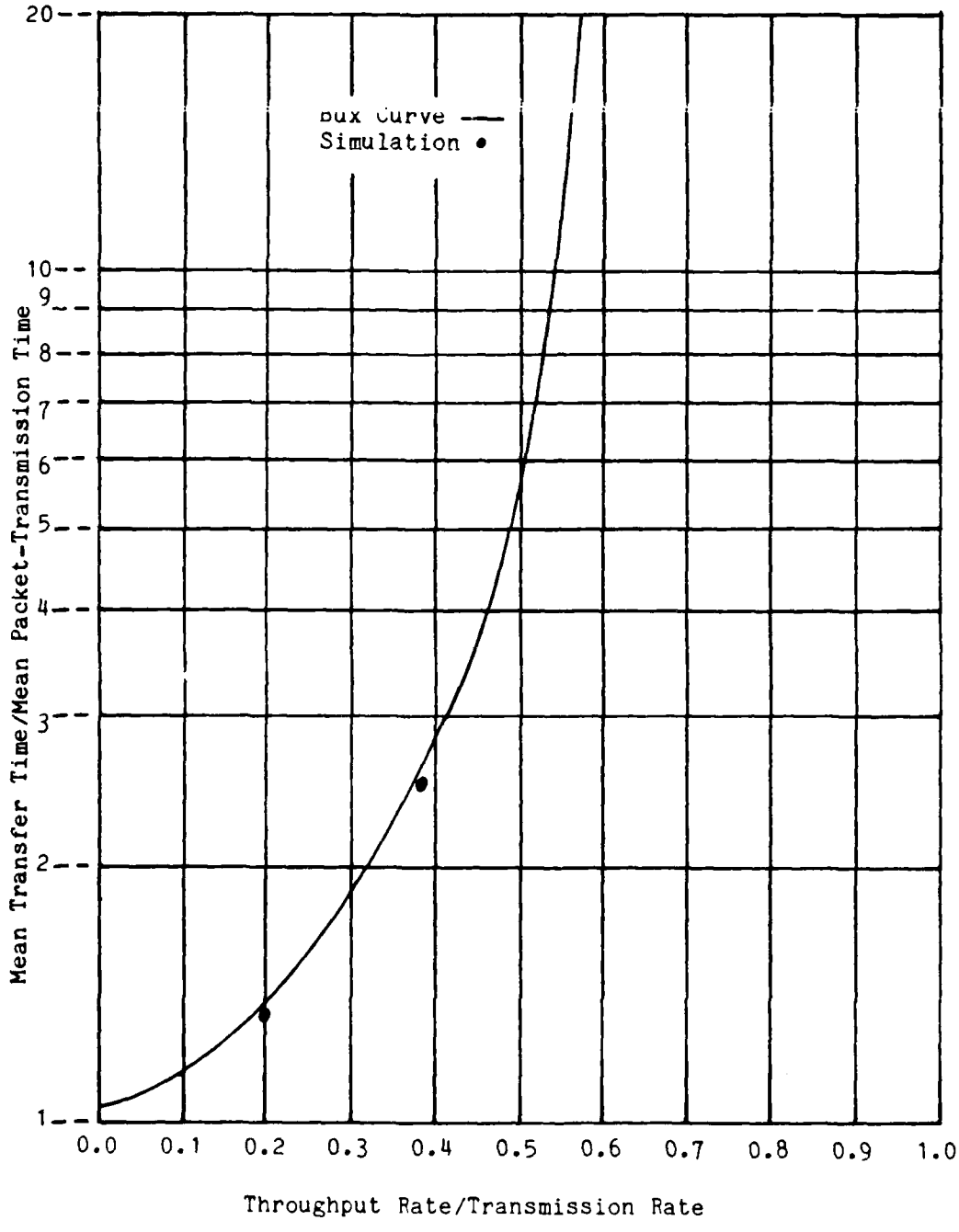


Figure 9. Transfer delay-throughput characteristics for a CSMA/CD at 10 MB/S.[14:1470]

The test setup used for the simulation model was:

TEST SETUP:

- Bus
  - -- 5 M bit/sec
  - -- 1-persistent
  - -- cable length 3280 ft
  - -- overhead 24 bits
  - -- cable delay 4.996 μsec
- Sessions
  - -- number of sessions 50
  - -- terminal rate 100000
  - -- input interval varied, +/- 0
  - -- number of characters per input 125

Calculated Results: The curve used by Bux for the 5 M bit/sec network is indicated by the solid line in Figure 10 on the next page.

Simulated Results: The results of various simulation runs are indicated by the plotted points in Figure 10.

Test Conclusion: Again, the simulation results basically follow the results used by Bux.

Summary

In this chapter, 5 tests were run to demonstrate the ability of the thesis simulation to model a CSMA/CD bus local area network. The results of test 3, 4, and 5 were compared with the analytical model of Bux. The comparison showed that the thesis simulation model does exhibit the

PART A2: Program Sessions

Comment. This program will allow you to update your list of typical sessions that you may wish to use in some future simulation. Having established a 'sessions' file, we will then call program Sessions to make some changes to the typical sessions file.

user's command: sessions

program's response:

The following is the current list of typical sessions.

ID	speed(b/s)	#char/input	input interval(msec)	input variance(msec)
1	9600	1	200	10

Do you wish to modify the current listing of typical sessions? (Y/N) <CR>

user' response : y <cr>

(comment: If 'n' had been answered, the program would have terminated. Any input other than a 'y' or 'n', would have repeated the question, do you wish to modify.)

program's response: Do you wish to make an addition or deletion? (A/D) <CR>

user's response: a <cr>

characters per input, input interval in msec, and input interval variance in msec. <CR>

user's response: 9600 1 200 10 <cr>

program's response: A new typical session file has been established. Any further changes should be done by executing program "sessions".

'program terminates'

PART A1: Program Input t(ypical sessions)

Comment. This program is normally used only once. It will create the file necessary to store typical sessions/customers that may be used in the future for simulation. Once the file has been created, program Sessions should be used to perform any necessary updating. This program could also be used if the user desired to completely erase the typical sessions file clean. If by some mistake the file named 'sessions' is destroyed, this program must be run to re-establish the 'sessions' file. The file 'sessions' is established by loading the first typical session into the file. Assuming that you have not created a 'sessions' file yet, we will then call program Input_t.

user's command: input_t

program's response: WARNING: This program should be used only to establish a new typical sessions file. If a typical sessions file presently exists, this program will erase that file and establish a new typical sessions file. Do you want to establish a new typical session file? (Y/N). <CR>

user's input: y <cr>

(comment: If you enter 'n' at this point, the program will simply terminate.)

program's respond: Please enter the terminal speed (b/s), number of

APPENDIX A: SIMULATION USER'S GUIDE

Introduction. Appendix A provides the user the necessary information to run the seven programs necessary to simulate a bus local area network utilizing either 1-persistent or p-persistent CSMA/CD. Appendix A is broken into seven parts, one part for each program. The seven parts and programs are:

- Part 1: Program Input_t(ypical sessons)
- Part 2: Program Sessions
- Part 3: Program Input_n(etwork sessions)
- Part 4: Program Network
- Part 5: Program Parameters
- Part 6: Program Simulate
- Part 7: Program Evaluate

These parts will take the user step-by-step through the operation of each program, with the final result being a simulation run. All user responses to the program prompts should be in integer values or a one character letter.

WARNING: The program Evaluate uses files generated by programs Network, Parameters, and Simulate. Therefore, users inputs to these programs should not be changed until after the program Evaluate has completed its run.

Program Input_t begins on the next page.

which prevents the simulation model from perfectly mimicking an actual CSMA/CD bus local area network. The chapter further attempts to show that these 6 known limitations have a minimal effect on the ability of the simulation model to model an actual LAN.

of the cable.

Limitation 5

Discussion: One major limitation of the present simulation working model is the restriction of the number of sessions to 500. This limitation is due solely to the size of the memory of the micro-computer being used.

Recommendation: If it is determined that more than 500 sessions are needed to be simulated, it is recommended that the Pascal computer programs be loaded into a computer with a larger memory capability.

Limitation 6

Discussion: Based on the built-in function 'in' of the Pascal language used, the maximum actual cable delay the simulation can use is 125 times the propagation delay of the cable. That is, if the actual delay of the bus cable is not known, the simulation defaults to the delay equal to the propagation delay of the cable being used. In the case of the 2000 ft HQ AFLC cable, the propagation delay is approximately 3.0 usec. Based on the way the function 'in' is used, the simulation model will accept an actual cable delay of 125 x 3 usec, or 375 usec.

Recommendation: If it is determined that the actual cable delay is greater than 125 times the propagation delay of the cable being used, then a new function or procedure must be used to replace the function 'in'.

Summary

The limitations discussed in this chapter, points out known areas

of the simulation clock is 3/131280 or .00002285, with a probabilty of no input of .99997715. Now using the 1000 customers and the Bernoulli Trials, the probability of no input for the 1000 customers is 97.74% and the probability of 1 input is 2.233%, with the probability of more than one input of .027%. Now assuming that each customer is inputing 80 characters per input, the amount of simulation time necessary for a 3 mega bits/sec bus to pass one input is 166. Assuming that one of the inputs is ready to transmit on the bus, the probability that another input will arrive at the bus while it is busy, is 166/131280 or .00126. Again using the remaining 999 customers and the Bernoulli Trials, the probability of no inputs reaching the bus while it is busy, is 28.38% and the probability of one session's input reaching the bus while it is busy, is 35.76%. In this particular case, the probability that more than one input will reach the bus while it is busy, is 35.86% and more than one customer waiting to use the bus will cause a collision. As can be seen from the figures, it is over a 1000 times more likely that a collision will occur because of traffic waiting for the bus, than because of inputs following closely behind each other.

Recommendation. The author's contention should hold for any local area networks which generates traffic from a large number of sources, at a relatively slow input rate. If the simulation model is to be used for simulation of a small number of sources with a relatively fast input rate, the error introduced by treating all collision the same, should be re-evaluated. Intuitively, it is felt that the error would not become excessive, since the time lost due to packets colliding is only a fraction of the total time lost because of collisions. Normally the amount of time waiting for re-transmission is much greater than the delay

cable, the average time, that would be lost due to collisions for sessions starting to transmit at the same time, is the time equating to 1/2 the delay of the cable.

In some cases, it is possible that one session will start to transmit and shortly before the input reaches the other end of the cable, someone else begins to transmit. In this particular case, the time lost due to collisions, can be as much as twice the round trip delay of the cable. So why isn't the average time lost due to colliding packets 1 delay of the cable instead of 1/2 the delay of the cable? As stated earlier in Chapter 4, the algorithm of the simulation model does take into account this case by insuring that no new inputs are made to a BIU transmitting data until the BIU has been transmitting for at least the time equal to two delay times of the cable. Also collisions are detected by scanning to see if another session will want to transmit within one delay time of the cable, from the time a session starts to transmit. Even though the simulation model allows for the possibility of a collision occurring because one input arrives at the bus a short time after another, the author contends that this occurrence is very rare and that most collisions occur because sessions try to transmit at approximately the same time. That is the reason, when collisions occur, that the simulation uses the average of 1/2 the delay of the cable for the amount of time lost due to colliding packets. To illustrate this contention, lets look at 1000 customers inputting data every 200 msec. Using the AFLC network for our example, the 200 msec will convert to 131280 ticks of the simulation clock. The time for the next input to be made can be anywhere between 1 and 131280 ticks of the simulation clock. Taking one customer first, the probabilty that an input will be made within the next 3 ticks

that a new pseudorandom number generator be found.

Limitation 3

Discussion: Limitation 3 is due to the fact that the expected type of traffic to be generated by the users was unknown. The +/- deviation of the interval between inputs represents an uniformly distributed deviation. It may be found in the future, that the input deviation is not uniform but normal or some other type of distribution.

Recommendation: If in the future, it is determined that the +/- uniform deviation of the interval between inputs is unacceptable, it is recommended that a function be written to weight the random numbers generated by the appropriate amount, to give the desired distribution needed.

Limitation 4

Discussion: Limitation 4 is that the simulation model treats all collisions the same. That is, when a collision occurs, a time interval equal to 1/2 the delay of the cable is added to the table summing up the delay time due to collisions. By doing this, the simulation model assumes the time lost due to colliding packets, is a fixed average amount. This average is determined from the following examples. If two sessions at opposite ends of a bus cable were to transmit at the same time, it would take the complete delay time of the cable before each knew there was a collision and stop transmitting. On the other hand, if two sessions side-by-side on the bus cable started to transmit at the same time, they would know almost instantaneously that a collision occurred and stop transmitting. With sessions uniformly distributed along the bus

for transmission, instead of 131280 ticks.

Limitation 2

Discussion: Limitation 2 occurs with the pseudorandom number generator function used by the program Simulate. A test of 10000 random numbers conducted by the writer of the random number function, indicates a .72% preference for higher numbers, as opposed to lower valued numbers. This .72% preference for higher numbers tends to skew sightly to the left, the +/- deviation of the interval between inputs. This skewing to the left occurs because random numbers are used to determine if the input deviation is + or -. Once the amount of the deviation is randomly picked, another random number is picked to determine if the deviation is + or -. The random numbers generated can appear anywhere between 1 and 65536. If the random number picked is greater than 32768, it becomes a negative deviation. Since the generator favors higher numbers slightly more, the average interval between inputs will be slightly less than the expected mean and the traffic passed by the terminals will be slightly higher than the expected mean. Some examples of the skewing effect are: an error of .018% is introduced for a 200 msec interval between inputs with a +/- 5 msec deviation, an error of .072% is introduced for a 200 msec interval with a +/- 20 msec deviation, and an error of .36% is introduced with a 200 msec interval with a +/- 100 msec deviation. The amount of error introduced depends on the size of the interval between inputs, in comparison to the size of the deviation. The larger the ratio of the deviation over the interval, the larger the error will be.

Recommendation: If the user feels that the accuracy of the pseudorandom number generator is unacceptable, then it is recommended

eight bit ASCII code word and the time it takes the BIU to add on the overhead to the information bits. These times were not available to the author since the terminals and BIUs to be used on the network were unknown. On the other hand, the author considers these times to be insignificant in the normal operation of the network. It can be imagined that these times would be only equal to a few microseconds or less (the time it takes a few shift registers to shift), and would only come into play if the terminal was passing data at its maximum capacity (such as a continuous stream of data at 9600 bits/sec). Normally the terminal will pass data at a set rate, such as 9600 bits/sec, and once the data has been passed, the terminal is idle. Limitation 1 only comes into play if the idle time between inputs is less than the time it takes to convert an input to an ASCII code word and the time it takes the BIU to add on overhead. An example of limitation 1 is; if the user was inputing data every 200 msec and 240 characters per input, and the time for conversion was 6 usec, the error caused by limitation 1 would only be .003%. And again, this error only occurs if the idle time between inputs is less than 6 usec.

Recommendation: If the user determines that the limitation needs to be eliminated, it can be done by a simple modification to the program Parameters. In program Parameters, the time it takes an input to travel from the terminal to the BIU is calculated. To eliminate limitation 1, it is only necessary to add an additional correction factor to this calculated value, in order to account for the delay caused by converting an input to an ASCII code word and the delay caused by adding overhead to the information bits. In the example given, it would now take the information 131284 ticks of the simulation clock for an input to be ready

VI. DISCUSSIONS AND RECOMMENDATIONS

Introduction

In Chapter 1, the need of HQ AFLC for a simulation model of a CSMA/CD bus LAN was discussed. It was also discussed that previously developed network models are very limited in the number of network variables they can handle. The problem was to develop a CSMA/CD bus simulation model which could handle the numerous variables associated with a real local area network. The thesis takes the approach that once an input is made, the time for the input to travel through the various stages of the network can be easily calculated. Therefore, the simulation model generates traffic based on the statistical parameters of each individual customer, then tracks the input as the simulation clock ticks. Using the memory power of the computer to keep track of the location of all inputs, the simulation model is able to determine the effect of an input on all other inputs. In some cases, an input has no effect on other inputs, and at the other extreme, when inputs want to use the bus at the same time, they have a drastic effect on each others performance. In Chapter 5 numerous tests were performed to demonstrate the ability of the simulation model to model a CSMA/CD bus LAN. The thesis simulation model attempts to mimic the characteristics of an actual LAN as close as possible. In the remaining portions of this chapter, various limitations of the simulation model to perfectly mimic a true LAN are discussed.

Limitation 1

Discussion: The first limitation of the simulation model is the time associated with the time it takes the terminal to convert an input to an

characteristics of a CSMA/CD bus LAN.

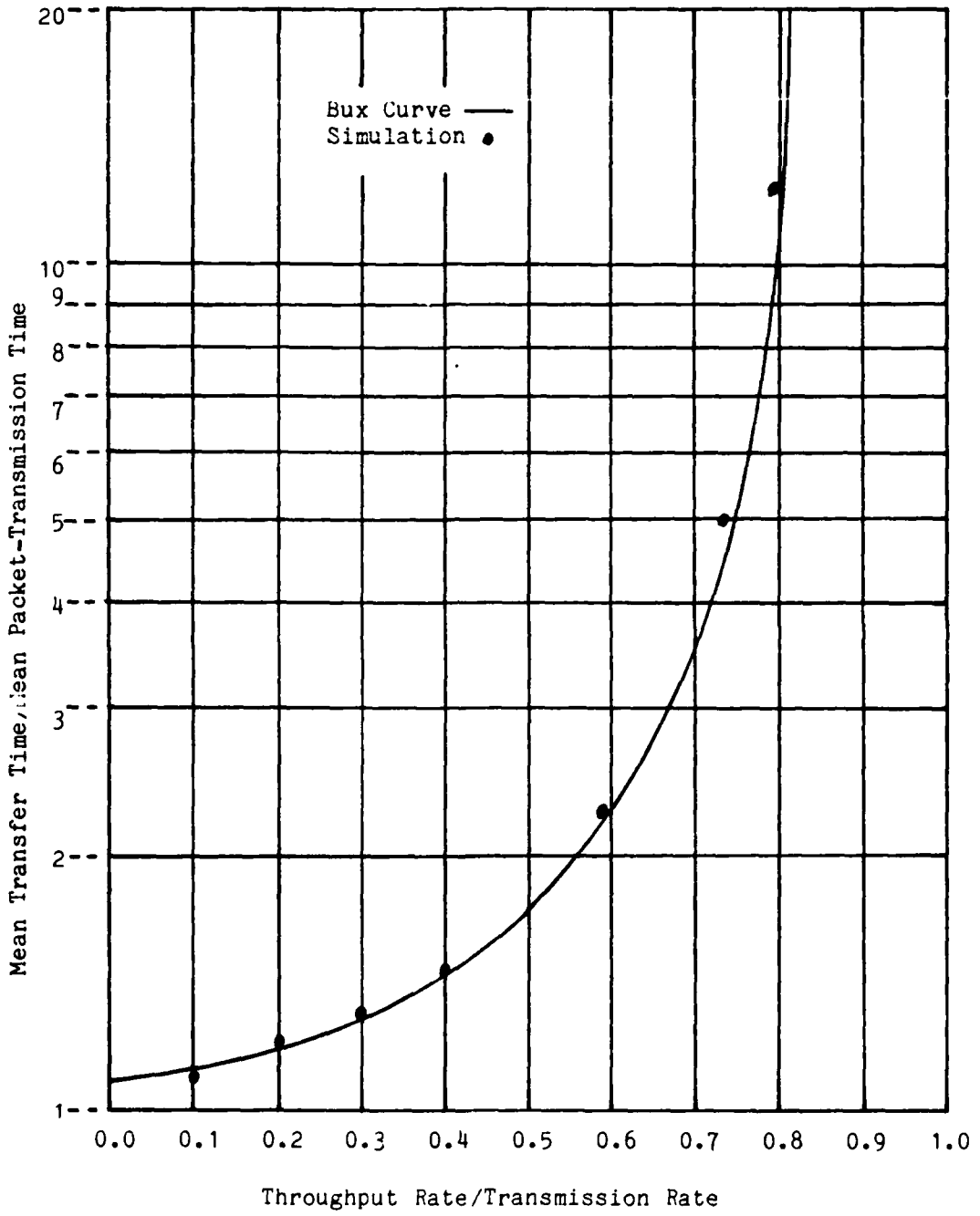


Figure 10. Transfer delay-throughput characteristics for a CSMA/CD at 5 MB/S.[14:1470]

program's response: Enter the data rate of the terminal (b/s), number of characters per input, the time interval between inputs (in msec), and the input variance (in msec) of the new session. <CR>

user's response: 19200 80 1000 10 <cr>

program's response: Please enter the ID number of the session you wish the new session to follow. <CR>

user's response: 1 <cr>

program's response:

The following is the current list of typical sessions.

ID	speed(b/s)	#char/input	input interval(msec)	input variance(msec)
1	9600	1	200	10
2	19200	80	1000	10

Do you want to make more changes? (Y/N) <CR>

user's response: y <cr>

program's response: Do you wish to make an addition or deletion? (Y/N) <CR>

user's response: d <cr>

program's response: Please enter the ID number of the session you wish to

delete. <CR>

user's response: 1 <cr>

program's response:

The following is the current list of typical sessions.

ID	speed(b/s)	#char/input	input interval(msec)	input variance(msec)
1	19200	80	1000	10

Do you want to make more changes? (Y/N) <CR>

user's response: n <cr>

'program terminates'

(comment: The cycle to make changes can be repeated as many times as you wish. The size of the file is only limited by the memory space of the computer being used.)

PART A3: Program Input n(etwork sessions)

Comment. This program, similar to input_t, is normally used only once. It establishes the file necessary to store sessions for simulation. Once the file has been created, program Network should be used to perform any necessary changes to the list of customers to be simulated. This program could also be used if the user desired to completely erase the network sessions file clean. If by some mistake the file named 'network' is destroyed, this program must be run to re-establish the 'network' file. The file 'network' is established by loading the first session to be simulated into the file. Assuming that you have not created a 'network' file yet, we will call program Input_n.

user's command: input_n

program's response: WARNING: This program should be used only to establish a new network sessions file. If a network sessions file presently exists, this program will erase that file and establish a new network sessions file. Do you want to establish a new network sessions file? (Y/N) <CR>

user's response: y <cr>

(comment: A 'n' response would have simply terminated the program.)

program's response: Please enter terminal speed(b/s), number of characters per input, input interval in msec, input interval variance in

msec, and the number of times you wish this session repeated in the simulation. <CR>

user's response: 9600 80 100 10 3 <cr>

program's response: A new network sessions file has been established. Any further changes to this file should be done by executing program Network.

'program terminates'

PART A4: Program Network

Comment: This program will allow you to update your list of customers/sessions you plan to simulate. Having established a 'network' file, we will then call program Network to make some changes to the network file.

user's command: network

program's response:

The following is the current list of network sessions.

ID	speed (b/s)	#char/input	input interval (msec)	input variance (msec)	quantity
2	9600	80	100	10	3
4	9600	80	50	1	2

Do you wish to make changes to the network sessions? (Y/N) <CR>

user's response: y <cr>

(comment: A 'n' response would have terminated the program.)

program's response: Do wish to make an addition or deletion? (A/D) <CR>

user's response: a <cr>

program's response: Do you need to see a copy of the current typical

sessions list before making your choice on your addition? (Y/N) <CR>

user's response: y <cr>

(comment: A 'n' response would have caused the program to by pass showing you the current file of typical customers.)

program's response:

The following is the current list of typical sessions.

ID	speed(b/s)	#char/input	input interval(msec)	input variance(msec)
1	19200	80	500	20
2	9600	80	100	10
3	9600	80	100	25
4	9600	80	50	1

Enter the ID number of the typical session you wish to add to the network. <CR>

user's response: 3 <cr>

program's response: Enter how many times do you wish to duplicate this session in the network. <CR>

user's response: 4 <cr>

program's response:

The following is the current list of network sessions.

ID	speed (b/s)	#char/input	input interval (msec)	input variance (msec)	quantity
2	9600	80	100	10	3
4	9600	80	50	1	2
3	9600	80	100	50	4

Do you want to make more changes? (Y/N) <CR>

user's response: y <cr>

program's response: Do you wish to make an addition or deletion? (A/D) <CR>

user's response: d <cr>

program's response: Please enter the ID of the network session you wish to delete. <CR>

user's response: 2 <cr>

program's response:

The following is the current list of network sessions.

ID	speed (b/s)	#char/input	input interval (msec)	input variance (msec)	quantity
4	9600	80	50	1	2
3	9600	80	100	50	4

Do you want to make more changes? (Y/N) <CR>

user's response: n

'program terminates'

PART A5: Program Parameters

Comment: Having established the list of sessions/customers to be simulated, the next step is to call program Parameters. This program will calculate the various travel times associated with each session to progress through the theoretical network. Bus characteristics will be loaded with this program. The user's responses indicated below are examples only. You should enter the particular information unique to your network.

user's command: parameters

program's response: Please enter the data rate of the bus (b/s). <CR>

user's response: 1000000 <cr>

program's response: Please enter, in feet, the round trip length of the bus cable. <CR>

user's response: 2000 <cr>

program's response: Please enter the overhead, in bits, of the BIU. <CR>

user's response: 120 <cr>

program's response: Enter the time of day in hours and minutes. (example: for 8:45 enter 845)

user's response: 1546 <cr>

(comment: The time of day is used only as a seed for the random number generator.)

program's response: Do you wish to see a display of the calculated data? (Y/N) <CR>

user's response: n <cr>

(comment: If 'y' is entered, the program will show a display of the calculated parameters before terminating.)

'program terminates'

(comment: In some cases the following note may be given by the program:

"**NOTE** The error in calculating the bus busy time for session #3 is 4%. The error is high because the bus busy time is approaching the basic unit of time for the simulator, one way propagation. One or more of the following factors may be contributing to the high error.

- high bus bite rate
- long length of bus cable
- low overhead
- low # of characters per input

Due to the high error, the results of the bus busy time indicated during

the data evaluation stage, should be ignored."

This note indicates that only one of the three methods used to calculate the amount of traffic being passed by the bus should be ignored. The three methods used to calculate the traffic being passed by the bus are discussed further in Chapter 5, Test 2.)

PART A6: Program Simulate

Comment. Up to this point, we have selected the customers/sessions we wish to simulate and have calculated the necessary parameters for the simulation run. Now it is time to perform the actual simulation. This program will prompt you for some last minute information prior to starting the simulation. Again, as before, user's responses are only examples and you should enter the information unique to your network.

user's command: simulate

program's response: (number of sessions being simulated) sessions were loaded for this simulation run. Do you wish to review the data loaded? (Y/N) <CR>

user's response: n <cr>

(comment: A 'y' input would have instructed the program to let you review the values calculated in the previous program Parameters and loaded for the simulation run.)

program's response: Is the actual round trip delay of the network known? (Y/N) <CR>

user's response: y

(comment: If 'n' was inputed, the program defaults and uses the basic

propagation delay of the cable length enter before, for the bus delay. A 'y' answer implies that you known the actual delay of bus cable due to propagation, amplifiers, splitters, etc.)

program's response: Please enter, in nano seconds, the known round trip delay of the bus. <CR>

user's response: 6000 <cr>

program's response: Please re-enter, in feet, the max round trip length of the bus cable. <CR>

user's response: 2000 <cr>

program's response:
Is this simulation a 1- or p- persistent?
Enter a 1 for a 1-persistent simulation or a 2 for a p-persistent simulation. <CR>

user's response: 1 <cr>

(comment: If a '2' is entered, the program will further prompt for the percentage between 1 and 100%.)

program's response: Please enter the transition clock time. <CR>

user's response: 250000 <cr>

program's response: Please enter the simulation clock time. <CR>

user's response: 10000000 <cr>

(comment: The times enter are with respect to the ticks of the simulation clock. One method for determining how long the simulation should run is to review the data loaded for simulation. Having reviewed the data, pick the session with the largest amount of time between inputs, then multiply that quantity by the number of inputs you wish to see simulated. The more inputs you allow a session to input, the better the accuracy of the simulation run.)

program's response: Please enter the time of day. (example: for 13:15 enter 1315) <CR>

user's response: 1548 <cr>

(comment: The time of day is used only as a seed for the random number generator.)

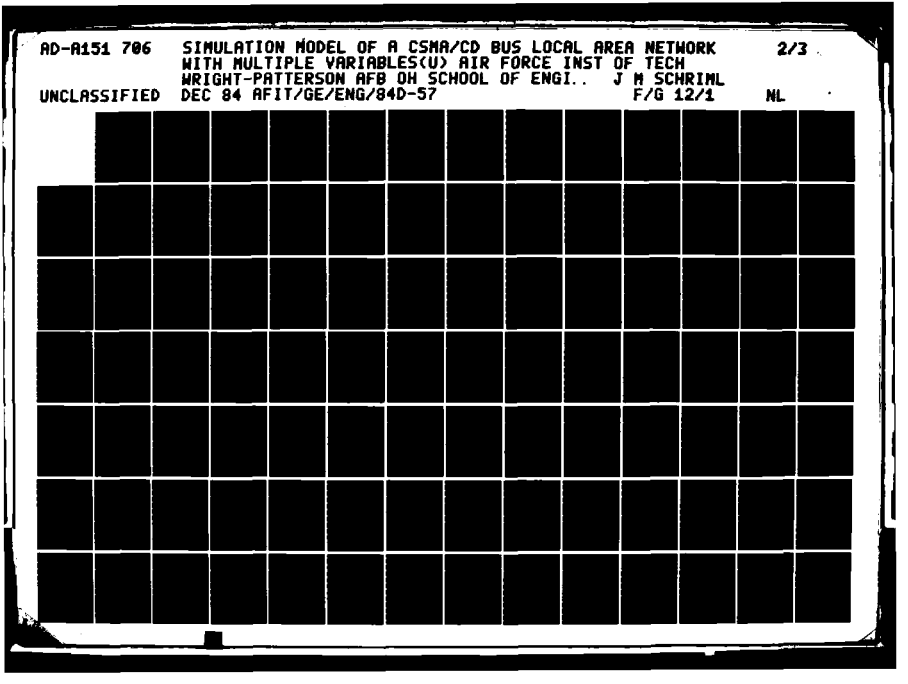
program's response: Do you need a display of the simulation run? (Y/N) <CR>

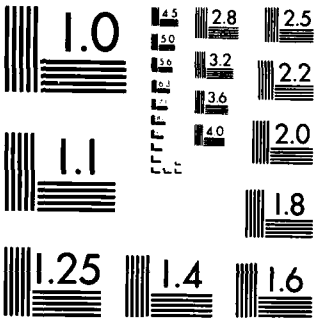
user's response: n <cr>

(comment: A 'y' answer will give you a display of every step the

AD-A151 706 SIMULATION MODEL OF A CSMA/CD BUS LOCAL AREA NETWORK WITH MULTIPLE VARIABLES(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J M SCHRIML 2/3

UNCLASSIFIED DEC 84 AFIT/GE/ENG/84D-57 F/G 12/1 NL





simulation performs. This option was originally used to help trouble shoot the program during the design stage. The option was left in to allow the user to actually watch the performance of a network in action. It is highly recommended, that if time allows, a few simulations using this option be performed. Real insight to the effects of different types traffic on network performance can easily be seen. Since there are no headings with this option, the list of functions for each column of the display is indicated below at the end of part 6.)

program's response: <Running clock time of the simulation counting down and the word 'collision' when collisions occur.>

program's response: Actual simulation clock is < time >.

'program terminates'

The following is a list of the functions of each column being displayed, if the option to watch the simulation run is used.

column 1 - This column keeps track of the time that must elapse before the traffic inputed by the terminal reaches the BIU. A zero value indicates no traffic and a one value indicates that the BIU is ready to transmit on the next tick of the clock.

column 2 - This column remains constant with the time that a particular session needs for its' traffic to travel from the terminal to the BIU.

column 3 - This column keeps track of the time that must elapse

before the next input is made. A zero value indicates that flow control is being employed. A one value indicates that an input is about to be made to the terminal.

column 4 - This column remains constant with the time interval between inputs for each particular session.

column 5 - This column remains constant with the time that the bus will be busy once a BIU is able to transmit for a particular session.

column 6 - This column sums up the waiting time a session's traffic must wait because the bus is busy.

column 7 - This column sums up the waiting time a session's inputs must wait because of flow control.

column 8 - This column sums up the number of collisions for each session. Only the row for the first session is used to record collisions, all other rows will remain at 0.

column 9 - This column sums up the number of inputs each session makes.

column 10 - This column sums the time the bus is busy for each session.

column 11 - This column sums up the extra waiting time a session's traffic must wait because of collisions.

column 12 - This column keeps track of the local variable needed for the exponential binary backoff.

column 13 - This column is used for storage of the amount of set back time due to a collision. As events occur, the length of time the event takes, is removed from storage and added to column 11.

column 14 - This column remains constant with the variance of the interval between inputs.

PART A7: Program Evaluate

Comment: Once the program Simulate is completed, the results of the simulation are filed away in the file 'simulate' for use by the program Evaluate. Having completed a simulation run, let's call the program Evaluate and review the performance of our network.

user's command: evaluate

program's response:

The results of the simulation run are based on the following network parameters.

Data rate of the bus was 1000000 bits/sec.

Round trip length of the bus cable 2000 feet.

Round trip delay of the cable 3046 nano seconds.

The BIU overhead was 120 bits.

1-persistent was used.

The actual transition clock time was 256293.

The actual data collecting time was 25000756.

The transition clock time approx equals 390 msec of real network run time.

The data collecting time approx equals 38087 msec of real network run time.

The performance of various types of sessions are as follows.

Please enter <CR> to continue.

user's response: <cr>

program's response:

10 typical sessions were set up as follows:

ID: 11

TERMINAL RATE: 9600 bits/sec

# of CHARACTERS PER INPUT: 80

TIME INTERVAL BETWEEN INPUTS: 50 msec var 0 msec

AVE MAX INPUT RATE: 12800 bits/sec

The perfomance was as follows.

DELAY PER INPUT:

  -due to traffic on the bus: 0 usec

  -due to collisions: 0 usec

  -due to flow control: 16668 usec

THROUGHPUT RATE PER SESSION:

  -time interval between inputs:

   --terminals indicate 66668 usec

   --inputs indicate 66668 usec

  -transmission rate:

   --terminals indicate 9599 bits/sec

   --inputs indicate 9599 bits/sec

Please enter <CR> to continue

user's response: <cr>

(comment: If additional types of sessions had been simulated, then a performance summation would have been displayed for each session type.)

program's response:

*** BUS PERFORMANCE ***

The bus is set up to operate at 1000000 bits/sec.

Traffic on the bus indicates a rate of 113799 bits/sec.

Traffic being passed by the terminals indicate a rate of 113997 bits/sec.

Traffic being inputed by the user indicates a bus rate of 113997 bits/sec.

Collisions were at the rate of 0 collisions per sec.

*** NETWORK INPUT TRAFFIC ***

The MAX traffic rate which the terminals can pass is 96000 bits/sec.

The simulation indicates that the terminals were passing traffic at 95998 bits/sec.

The users are attempting to input 128000 bits/sec.

The simulation indicates that the network is allowing an input of 95998 bits/sec.

Please enter <CR> to continue.

user's response: <cr>

program's response: Do you wish to review the raw data? (Y/N) <CR>

user's response: n <cr>

(comment: A 'y' would have displayed a copy of the raw data collected by the simulation run before terminating the program.)

'program terminates'

APPENDIX B: SIMULATION COMPUTER PROGRAMS

Introduction. Appendix B provides the actual seven Pascal computer programs written to simulate a bus local area network utilizing either 1-persistent or p-persistent CSMA/CD. The programs are centered around a Zenith Z-100 micro-computer using the 16 bit MS/DOS Pascal Language. The micro-computer's 128K memory limits the number of simulation customers/sessions to 500. By making the minor modification necessary to make the Pascal programs compatible with a computer with larger memory, the number of sessions can be greatly extended. The computer programs were written using only standard Pascal language. The only anticipated modifications, necessary to use these Pascal programs on other types of computers, are how the version of the Pascal language being used, handles external files and integers. The integer value allowed in these programs range between +/- 32768. The version of the Pascal language being used in this thesis also allows for an extended integer called integer4. The integer4 values range between +/- 2147483648.

Appendix B is broken into seven parts, one part for each program. The identifiers used as variables were picked with the hope that they would make it easier for future readers to follow and understand the programs. To assist the reader even further, a list of variables used and their purpose are given prior to each Pascal program. A variable followed by a '(G)' indicates a global variable while a '(L)' indicates a local variable. The second character indicates the type of variable, if applicable; '(I)' indicates integer, '(I4)' indicates integer4, '(C)' indicates character, and '(R)' indicates real. The first Pascal program begins on the next page.

PART B1: Pascal Program Input-t(ypical sessions)

The variables and their purpose, as used with the program Input_t, are as follows:

VARIABLES:

answer (G)(C) - Represents the user's response to a question.

datafile (G) - Represents a file of records containing typical sessions.

session (G) - Represents a record of a typical session.

session.id (G)(I) - Represents the ID number of the typical session.

session.interval (G)(I4) - Represents in msec the time interval between inputs.

session.number (G)(I4) - Represents the number of characters inputed by each input.

session.speed (G)(I4) - Represents in bits/sec the data rate of the session's terminal.

session.variance (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

The actual Pascal program, Input_t, begins on the next page.

```
{* PROGRAM:INPUT_T(YPICAL_SESSIONS) *}

{************************************************************************}
{  DATE: 29 Aug 84                                                       }
{  VERSION: 1.0                                                          }
{  NAME: input_t(ypical_sessions)                                        }
{  FUNCTION:  This program will establish a new typical session file.    }
{             A typical session file is a file which contains various    }
{             types of sessions which may be used in the simulation      }
{             of the local area network.  If a typical session file      }
{             already exists, this program will erase that file and      }
{             create a new typical session file.                         }
{  INPUTS: none                                                          }
{  OUTPUTS: none                                                         }
{  GLOBAL VARIABLES: answer, session, datafile                           }
{  GLOBAL TABLES USED: none                                              }
{  GLOBAL TABLES CHANGED: none                                           }
{  FILES READ: none                                                      }
{  FILES WRITTEN: sessions                                               }
{  PROCEDURES CALLED: none                                               }
{  CALLING PROCEDURES: none                                              }
{  AUTHOR: Capt John M. Schriml                                          }
{  HISTORY: none                                                         }
{************************************************************************}


program input_t (input,output);

  type typical_session = record
                           id:integer;
                           speed,number,interval,variance:integer4
                         end;

  var session:typical_session;
      datafile:file of typical_session;
      answer:char;

  begin
    writeln;
    write ('WARNING: This program should be used only to establish a ');
    writeln ('new typical session ');
    write ('file.  If a typical session file ');
    writeln ('presently exists, this program will erase that ');
    writeln ('file and establish a new typical session file.');
    writeln;
    writeln ('Do you want to establish a new typical session file?',
             ' (Y/N)');
    writeln ('<CR>');
    read (answer);
    writeln;
    writeln;
```

```
  if (answer = 'y') or (answer ='Y') then
    begin
      assign (datafile,'sessions');
      rewrite (datafile);
      write ('Please enter the terminal speed(b/s), number of ');
      writeln ('chararters per input, input ');
      write ('interval in msec, and input ');
      writeln ('interval variance in msec.<CR>');
      read (session.speed,session.number,session.interval);
      readln (session.variance);
      session.id := 1;
      datafile^ := session;
      put (datafile);
      close (datafile);
      writeln;
      write ('A new typical session file has been established. Any ');
      writeln ('further changes to this');
      write ('new file should be done by executing program ');
      writeln ('"sessions".')
    end
end.
```

PART B2: Program Sessions

The variables and their purpose, as used with the program Sessions, are as follows:

VARIABLES:

ans (G)(C) - Represents the user's reponse to a question.

answer (G)(C) - Represents the user's reponse to a question.

counter (L)(I) - Is used as a counter to renumber the typical sessions.

datafile (G) - Represents a file of records containing typical sessions.

datatemporary (G) - Represents a temporary file of records containing typical sessions.

remove (L)(I) - Represents the ID number of the session to be removed from the file of typical sessions.

session (G) - Represents the record of a typical session.

session.id (G)(I) - Represents the ID number of a typical session.

session.interval (G)(I4) - Represents in msec the time interval between inputs.

session.number (G)(I4) - Represents the number of characters inputed by each input.

session.speed (G)(I4) - Represents in bits/sec the data rate of the session's terminal.

session.variance (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

temporary session (G) - Represents a temporary record of a typical

session.

temporary session.id (G)(I) - Represents the temporary ID number of a typical session.

temporary session.interval (G)(I4) - Represents in msec the temporary time interval between inputs.

temporary session.number (G)(I4) - Represents the temporary number of characters inputed by each input.

temporary session.speed (G)(I4) - Represents in bits/sec the temporary data rate of the sessons's terminal.

temporary session.variance (G)(I4) - Represents in msec the temporary maximum deviation of the interval between inputs.

The actual Pascal program, Sessions, begins on the next page.

```
{  OUTPUTS: none                                                              }
{  GLOBAL VARIABLES: T_session, datafile                                      }
{  GLOBAL TABLES USED: none                                                   }
{  GLOBAL TABLES CHANGED: none                                                }
{  FILES READ: sessions                                                       }
{  FILES WRITTEN: none                                                        }
{  PROCEDURES CALLED: none                                                    }
{  CALLING PROCEDURES: addition                                               }
{  AUTHOR: Capt John M. Schriml                                               }
{  HISTORY: none                                                              }
{*************************************************************************}


  procedure display_typical_sessions;

    begin
      writeln;
      writeln('The following is the current list of typical sessions.');
      writeln;
      writeln;
      write (' id    speed    #char/input   ');
      writeln ('input interval(msec)   input variance(msec)');
      reset (datafile);
      while not eof (datafile) do
        begin
          T_session := datafile^;
          get (datafile);
          write (T_session.id:3,T_session.speed:9,T_session.number:12);
          writeln (T_session.interval:17,T_session.variance:24)
        end;
      close (datafile)
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                            }
{  VERSION: 1.0                                                               }
{  NAME: display_network_sessions                                             }
{  FUNCTION: Displays the current sessions that will be used in the           }
{            simulation of the local area network.                            }
{  INPUTS: none                                                               }
{  OUTPUTS: none                                                              }
{  GLOBAL VARIABLES: N_session, netfile                                       }
{  GLOBAL TABLES USED: none                                                   }
{  GLOBAL TABLES CHANGED: none                                                }
{  FILE READ: network                                                         }
{  FILES WRITTEN: none                                                        }
{  PROCEDURES CALLED: none                                                    }
{  CALLING PROCEDURES: addition, delete                                       }
{  AUTHOR: Capt John M. Schriml                                               }
{  HISTORY: none                                                              }
{*************************************************************************}
```

```
                {* PROGRAM:NETWORK *}

{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: network                                                          }
{  FUNCTION: This program gives the user access to the file which         }
{            contains the list of sessions that will be used in the       }
{            simulation of the local area network.  The user has the      }
{            option to add or delete sessions from the network session    }
{            file.                                                        }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: ans(wer)                                             }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: display_network_sessions                            }
{  CALLING PROCEDURES: none                                               }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


program network (input,output);

  type typical_session = record
                           id:integer;
                           speed,number,interval,variance:integer4
                         end;

       network_session  = record
                           id:integer;
                           speed,number,interval,variance,
                           quantity:integer4
                         end;

  var T_session,T_temporary_session:typical_session;
      N_session,N_temporary_session:network_session;
      datafile:file of typical_session;
      netfile,nettemporary:file of network_session;
      answer,ans:char;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: display_typical_sessions                                         }
{  FUNCTION: Displays a copy of the typical session file.                 }
{  INPUTS: none                                                           }
```

T-session.number (G)(I4) - Represents the number of characters inputed by each input.

T-session.speed (G)(I4) - Represents in bits/sec the data rate of the session's terminal.

T-session.variance (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

T-temporary session (G) - Represents a temporary record of a session.

T-temporary session.id (G)(I) - Represents the ID number of a session.

T-temporary session.interval (G)(I4) - Represents in msec the time interval between inputs.

T-temporary session.number (G)(I4) - Represents the number of characters inputed by each input.

T-temporary session.speed (G)(I4) - Represents in bits/sec the data rate of the session's terminal.

T-temporary session.variance (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

The actual Pascal program, Network, begins on the next page.

N-session.speed (G)(I4) - Represents in bits/sec the data rate of the session's terminal.

N-session.variance (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

N-temporary session (G) - Represents a temporary record of a network session.

N-temporary session.id (G)(I) - Represents the ID number of the network session.

N-temporary session.interval (G)(I4) - Represents in msec the interval between inputs.

N-temporary session.number (G)(I4) - Represents the number of characters inputed by each input.

N-temporary session.quantity (G)(I4) - Represents the number of times a session is to be repeated in a simulation.

N-temporary session.speed (G)(I4) - Represents in bits/sec the data rate of the session's terminal.

N-temporary session.variance (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

remove (L)(I) - Represents the ID number of the session being removed from the network file.

repetitions (L)(I4) - Represents the number of times a session is to be repeated in a simulation.

T-session (G) - Represents a record of a typical session.

T-session.id (G)(I) - Represents the ID number of the typical session.

T-session.interval (G)(I4) - Represents in msec the time interval between inputs.

PART B4: Pascal Program Network

The variables and their purpose, as used with the program Network, are as follows:

VARIABLES:

add (L)(I) - Represents the ID number of the typical session to be added to the network for simulation.

ans (G)(C) - Represents the user's response to a question.

answer (G)(C) - Represents the user's response to a question.

check (L)(I) - Is used to insure a valid ID number was entered by the user.

datafile (G) - Represents a file of records containing typical sessions.

netfile (G) - Represents a file of records containing typical sessions that were selected for simulation.

nettemporary (G) - Represents a temporary file of records containing the typical sessions selected for simulation.

N-session (G) - Represents a record of a network session.

N-session.id (G)(I) - Represents the ID number of the network session.

N-session.interval (G)(I4) - Represents in msec the interval between inputs of a network session.

N-session.number (G)(I4) - Represents the number of characters inputed by each input.

N-session.quantity (G)(I4) - Represents the number of times a session is to be repeated in the simulation network.

```
  writeln;
  if (answer = 'y') or (answer = 'Y') then
    begin
      assign (datafile,'network');
      rewrite (datafile);
      write ('Please enter terminal speed(b/s), number of ');
      writeln ('characters per input, input ');
      write ('interval in msec, input ');
      writeln('interval variance in msec, and quantity of session.');
      writeln ('<CR>');
      read (session.speed,session.number,session.interval);
      readln (session.variance,session.quantity);
      session.id := 1;
      datafile^ := session;
      put (datafile);
      close (datafile);
      writeln;
      write('A new network session file has been established. Any ');
      writeln ('further changes to this');
      write ('new file should be done by ');
      writeln ('executing program "network".')
    end
end.
```

```
                    {* PROGRAM:INPUT_N(ETWORK SESSIONS) *}

{************************************************************************}
{  DATE: 29 AUG 84                                                       }
{  VERSION: 1.0                                                          }
{  NAME: input_n(etwork sessions)                                        }
{  FUNCTION: This program will establish a new network session file.     }
{            A network session file is a file of the sessions that       }
{            will be used in the simulation of the local area network.   }
{            If a network session file already exists, this program      }
{            will erase that file and create a new typical network       }
{            file.                                                       }
{  INPUTS: none                                                          }
{  OUTPUTS: none                                                         }
{  GLOBAL VARIABLES: answer, session, datafile                           }
{  GLOBAL TABLES USED: none                                              }
{  GLOBAL TABLES CHANGED: none                                           }
{  FILES READ: none                                                      }
{  FILES WRITTEN: network                                                }
{  PROCEDURES CALLED: none                                               }
{  CALLING PROCEDURE: none                                               }
{  AUTHOR: Capt John M. Schriml                                          }
{  HISTORY: none                                                         }
{************************************************************************}


program input_n (input,output);

  type network_session = record
                           id:integer;
                           speed,number,interval,variance,
                           quantity:integer4
                           end;

  var session:network_session;
      datafile:file of network_session;
      answer:char;

  begin
    writeln;
    write ('WARNING: This program should be used only to establish ');
    writeln ('a new network session ');
    write ('file. If a network session file ');
    writeln ('presently exists, this program will erase that ');
    writeln ('file and establish a new network session file.');
    writeln;
    write ('Do you want to establish a new network session file? ');
    writeln ('(Y/N)');
    writeln ('<CR>');
    read (answer);
    writeln;
```

PART B3: Pascal Program Input-n(etwork sessions)

The variables and their purpose, as used in the program Input_n, are as follows:

VARIABLES:

answer (G)(C) - Represents the user's response to a question.

datafile (G) - Represents a file of records containing network sessions.

session (G) - Represents a record of a network session.

session.id (G)(I) - Represents the ID number of the typical session being used in the simulation run.

session.interval (G)(I4) - Represents in msec the time interval between inputs.

session.number (G)(I4) - Represents the number of characters inputed by each input.

session.quantity (G)(I4) - Represents the number of times to repeat a typical session in the simulation network.

session.speed (G)(I4) - Represents in msec the data rate of the session's terminal.

session.variance (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

The actual Pascal program, Input_n, begins on the next page.

```
      writeln
    until (answer ='n') or (answer ='N')
  end;


begin
  assign (datafile,'sessions');
  assign (datatemporary,'datatemp');
  display;
  repeat
    writeln;
    write ('Do you wish to modify the current listing of typical');
    writeln (' sessions. (Y/N)');
    writeln ('<CR>');
    readln (ans);
    writeln;
    writeln
  until (ans = 'Y') or (ans ='y') or (ans = 'n') or (ans ='N');
  if (ans = 'Y') or (ans = 'y')
    then modify
end.
```

```
          temporary_session.id := counter;
          datafile^ := temporary_session;
          put (datafile)
        end;
      close (datafile);
      close (datatemporary);
      display
    end;


{*************************************************************************}
{  DATE: 29 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: modify                                                           }
{  FUNCTION: This procedure prompts the user for what type of changes     }
{            they wish to make to the typical session file and then       }
{            calls the appropriate procedure.                             }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: answer                                               }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: addition, delete                                    }
{  CALLING PROCEDURES: sessions                                           }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure modify;

    begin
      repeat
        writeln ('Do you wish to make an addition or deletion? (A/D)');
        writeln ('<CR>');
        readln (answer);
        writeln;
        if (answer = 'a') or (answer = 'A')
          then addition
          else if (answer = 'd') or (answer = 'D')
            then delete
            else
              begin
                writeln;
                writeln ('An invalid character was entered.')
              end;
        writeln;
        writeln ('Do you want to make more changes? (Y/N)');
        writeln ('<CR>');
        readln (answer);
        writeln;
```

```
{************************************************************************}
{  DATE: 29 Aug 84                                                       }
{  VERSION: 1.0                                                          }
{  NAME: delete                                                          }
{  FUNCTION: Allows the user to delete a typical session from the        }
{            typical session file.                                       }
{  INPUTS: none                                                          }
{  OUTPUTS: none                                                         }
{  GLOBAL VARIABLES: none                                                }
{  GLOBAL TABLES USED: none                                              }
{  GLOBAL TABLES CHANGED: none                                           }
{  FILES READ: sessions, datatemp                                        }
{  FILES WRITTEN: sessions,datatemp                                      }
{  PROCEDURES CALLED: display                                            }
{  CALLING PROCEDURES: modify                                            }
{  AUTHOR: Capt John M. Schriml                                          }
{  HISTORY: none                                                         }
{************************************************************************}


  procedure delete;

    var remove,counter:integer;

    begin
      writeln;
      writeln ('Please enter the id of the session you wish to delete.',
               '<CR>');
      readln (remove);
      rewrite (datatemporary);
      reset (datafile);
      while not eof (datafile) do
        begin
          session := datafile^;
          get (datafile);
          if session.id <> remove then
            begin
              datatemporary^ :=session;
              put (datatemporary)
            end
        end;
      close (datafile);
      close (datatemporary);
      rewrite (datafile);
      reset (datatemporary);
      counter := 0;  {The counter is used to renumber the sessions.}
      while not eof (datatemporary) do
        begin
          temporary_session := datatemporary^;
          get (datatemporary);
          counter := counter + 1;
```

```
begin
  writeln;
  write ('Enter the data rate of the terminal(b/s), ');
  writeln ('number of characters per input,');
  writeln ('the time interval between inputs (in msec),',
           ' and the input interval');
  writeln ('variance (in msec) of the new session.');
  writeln ('<CR>');
  read (temporary_session.speed,temporary_session.number);
  readln (temporary_session.interval,temporary_session.variance);
  write ('Please enter id number of the session you wish the');
  writeln (' new session to follow.');
  writeln ('<CR>');
  readln (temporary_session.id);
  writeln;
  rewrite (datatemporary);
  reset (datafile);
  while not eof (datafile) do
    begin
      session := datafile^;
      get (datafile);
      if temporary_session.id = session.id then
        begin
          datatemporary^ := session;
          put (datatemporary);
          datatemporary^ := temporary_session;
          put (datatemporary)
        end
      else
        begin
          datatemporary^ :=session;
          put (datatemporary)
        end
    end;
  close (datafile);
  close (datatemporary);
  rewrite(datafile);
  reset (datatemporary);
  counter := 0;   {The counter is used to renumber the sessions}
  while not eof (datatemporary) do
    begin
      temporary_session := datatemporary^;
      get (datatemporary);
      counter := counter + 1;
      temporary_session.id := counter;
      datafile^ := temporary_session;
      put (datafile)
    end;
  close (datafile);
  close (datatemporary);
  display
end;
```

```
{  CALLING PROCEDURE: program sessions, addition, delete                }
{  AUTHOR: Capt John M. Schriml                                         }
{  HISTORY: none                                                        }
{***********************************************************************}


  procedure display;

    begin
      writeln;
      write ('The following is the current list of typical ');
      writeln ('sessions.');
      writeln;
      writeln;
      write (' id    speed(b/s)   #char/input   ');
      writeln ('input interval(msec)     input variance(msec)');
      writeln;
      reset (datafile);
      while not eof (datafile) do
        begin
          session := datafile^;
          get (datafile);
          write (session.id:3,session.speed:11,session.number:14);
          writeln (session.interval:16,session.variance:25);
        end;
      close(datafile)
    end;


{***********************************************************************}
{  DATE: 29 Aug 84                                                      }
{  VERSION: 1.0                                                         }
{  NAME: addition                                                       }
{  FUNCTION: Allows the user to make additions to the file of typical   }
{            sessions.                                                  }
{  INPUTS: none                                                         }
{  OUTPUTS: none                                                        }
{  GLOBAL VARIABLES: session, temporary_session, datafile,              }
{                    datatemporary                                      }
{  GLOBAL TABLES USED: none                                             }
{  GLOBAL TABLES CHANGED: none                                          }
{  FILES READ: sessions, datatemp                                       }
{  FILES WRITTEN: sessions, datatemp                                    }
{  PROCEDURES CALLED: display                                           }
{  CALLING PROCEDURES: modify                                           }
{  AUTHOR: Capt John M. Schriml                                         }
{  HISTORY: none                                                        }
{***********************************************************************}


  procedure addition;

    var counter:integer;
```

```
{* PROGRAM:SESSIONS *}


{************************************************************************}
{  DATE: 29 Aug 84                                                       }
{  VERSION: 1.0                                                          }
{  NAME: sessions                                                        }
{  FUNCTION: This program gives the user access to the file which        }
{            contains a list of typical sessions that may be selected    }
{            for use in the simulation of the local area network.  The   }
{            user has the option to add or delete sessions from the      }
{            typical session file.                                       }
{  INPUTS: none                                                          }
{  OUTPUTS: none                                                         }
{  GLOBAL VARIABLES: ans(wer)                                            }
{  GLOBAL TABLES USED: none                                              }
{  GLOBAL TABLES CHANGED: none                                           }
{  FILES READ: none                                                      }
{  FILES WRITTEN: none                                                   }
{  PROCEDURE CALLED: modify                                              }
{  CALLING PROCEDURE: none                                               }
{  AUTHOR: Capt John M. Schriml                                          }
{  HISTORY: none                                                         }
{************************************************************************}


program sessions (input,output);

  type typical_session = record
                           id:integer;
                           speed,number,interval,variance:integer4
                         end;

  var session,temporary_session:typical_session;
      datafile,datatemporary:file of typical_session;
      answer,ans:char;


{************************************************************************}
{  DATE: 29 Aug 84                                                       }
{  VERSION: 1.0                                                          }
{  NAME: display                                                         }
{  FUNCTION: Displays the current file of typical sessions.              }
{  INPUTS: none                                                          }
{  OUTPUTS: none                                                         }
{  GLOBAL VARIABLES: session, datafile                                   }
{  GLOBAL TABLES USED: none                                              }
{  GLOBAL TABLES CHANGED: none                                           }
{  FILES READ: sessions                                                  }
{  FILES WRITTEN: none                                                   }
{  PROCEDURES CALLED: none                                               }
```

```
procedure display_network_sessions;

  begin
    writeln;
    writeln('The following is the current list of network sessions.');
    writeln;
    writeln;
    write (' id   speed   #char/input   input interval(msec)');
    writeln ('  input variance(msec)  quantity');
    writeln;
    reset (netfile);
    while not eof (netfile) do
      begin
        N_session := netfile^;
        get (netfile);
        write (N_session.id:3,N_session.speed:8);
        write (N_session.number:10,N_session.interval:20);
        writeln (N_session.variance:21,N_session.quantity:16);
      end;
    close (netfile)
  end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: addition                                                         }
{  FUNCTION: Allows the user to copy a typical session from the           }
{            typical session file and place it into the file of           }
{            network sessions.                                            }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: T_session, N_temporary_session, datafile,            }
{                    netfile, nettemporary                                }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: sessions, network, nettemp                                 }
{  FILES WRITTEN: network, nettemp                                        }
{  PROCEDURES CALLED: display_network_sessions                            }
{  CALLING PROCEDURES: modify                                             }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


procedure addition;

  var add,check:integer;
      repetitions:integer4;

  begin
    writeln;
```

```
write ('Do you need to see a copy of the current typical ');
writeln ('sessions list before making');
writeln ('your choice on your addition.(Y/N)');
writeln ('<CR>');
readln (answer);
writeln;
if (answer = 'y') or (answer = 'Y')
  then display_typical_sessions;
writeln;
write ('Enter the id number of the typical sessions you wish ');
writeln ('to add to the network.');
writeln ('<CR>');
readln (add);
write ('Enter how many times you wish to duplicate this ');
writeln ('session in the network.');
writeln ('<CR>');
readln (repetitions);
reset (datafile);
check := 0;
while not eof (datafile) do
  begin
    T_session := datafile^;
    get (datafile);
    if add = T_session.id then
      begin
        N_temporary_session.id := T_session.id;
        N_temporary_session.speed := T_session.speed;
        N_temporary_session.number := T_session.number;
        N_temporary_session.interval := T_session.interval;
        N_temporary_session.variance := T_session.variance;
        N_temporary_session.quantity := repetitions;
        check := 1
      end
  end;
close (datafile);
rewrite (nettemporary);
reset (netfile);
while not eof (netfile) do
  begin
    N_session := netfile^;
    get (netfile);
    nettemporary^ := N_session;
    put (nettemporary)
  end;
if check = 1 then
  begin
    nettemporary^ := N_temporary_session;
    put (nettemporary)
  end;
close (nettemporary);
close (netfile);
rewrite (netfile);
reset (nettemporary);
```

```
      while not eof (nettemporary) do
        begin
          N_temporary_session := nettemporary^;
          get (nettemporary);
          netfile^ := N_temporary_session;
          put (netfile)
        end;
      close (netfile);
      close (nettemporary);
      display_network_sessions
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: delete                                                           }
{  FUNCTION: Allows the user to remove a session from the network         }
{            session file.                                                }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: netfile, nettemporary, N_session                     }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: network, nettemp                                           }
{  FILES WRIITEN: network, nettemp                                        }
{  PROCEDURES CALLED: display_network_sessions                            }
{  CALLING PROCEDURES: modify                                             }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure delete;

    var remove:integer;

    begin
      writeln;
      write ('Please enter the id of the network session you wish ');
      writeln ('to delete.');
      writeln ('<CR>');
      readln (remove);
      rewrite (nettemporary);
      reset (netfile);
      while not eof (netfile) do
        begin
          N_session := netfile^;
          get (netfile);
          if N_session.id <> remove then
            begin
              nettemporary^ := N_session;
              put (nettemporary)
```

```
          end
        end;
      close (netfile);
      close (nettemporary);
      rewrite (netfile);
      reset (nettemporary);
      while not eof (nettemporary) do
        begin
          N_temporary_session := nettemporary^;
          get (nettemporary);
          netfile^ := N_temporary_session;
          put (netfile)
        end;
      close (netfile);
      close (nettemporary);
      display_network_sessions
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: modify                                                           }
{  FUNCTION: This procedure prompts the user for what type of changes     }
{            they wish to make to the network session file and then       }
{            calls the appropriate procedure.                             }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: answer                                               }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: addition, delete                                    }
{  CALLING PROCEDURES: network                                            }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure modify;

    begin
      repeat
        writeln('Do wish to make an addition or deletion? (A/D)');
        readln (answer);
        writeln;
        if (answer = 'A') or (answer = 'a')
          then addition
          else if (answer = 'd') or (answer = 'D')
            then delete
            else
              begin
```

```
                writeln;
                writeln ('An invalid character was entered.')
              end;
        writeln;
        writeln ('Do you want to make more changes? (Y/N)');
        readln (answer);
        writeln;
        writeln
      until (answer ='n') or (answer ='N')
    end;


begin
  assign (datafile,'sessions');
  assign (netfile,'network');
  assign (nettemporary,'nettemp');
  display_network_sessions;
  repeat
    writeln;
    write ('Do you wish to make changes to the network ');
    writeln ('sessions. (Y/N)');
    readln (ans);
    writeln;
    writeln
  until (ans ='y') or (ans ='Y') or (ans ='n') or (ans = 'N');
  if (ans = 'Y') or (ans = 'y')
    then modify
end.
```

PART B5: Pascal Program Parameters

The variables and their purpose, as used with the program Parameters, are as follows:

VARIABLES:

answer (G)(C) - Represents the user's response to a question.

bus (G) - Represents a record of the bus information inputed by the user.

bus.bus-one-way-length (G)(I4) - Represents in feet the one way length of the bus cable.

bus.bus-overhead (G)(I4) - Represents the number of management bits (overhead) added by the BIU to the user's information bits prior to transmitting on the bus.

bus.bus-speed (G)(I4) - Represents in bits/sec the data rate of the bus cable.

busfile (G) - Represents the file that contains the bus information inputed by the user. File is used later by program Evaluate.

bus-length (G)(I) - Represents in feet the one way length of the bus cable.

bus-rate (G)(I4) - Represents in bits/sec the data rate of the bus cable.

datafile (G) - Represents the file of network sessions that will be simulated.

overhead (G)(I) - Represents the number of management bits (overhead) added by the BIU to the user's information bits prior to transmitting.

parameters (G) - Represents the file of records containing the times for a session's input to travel though a theoretical network.

range (G)(I) - Represents the number of sessions to be simulated. Since the number of sessions to be simulated is unknown prior to a simulation, range counts the number of sessions inputed and is used when loops are involved.

recipical-delay (G)(R) - Represents the recipical of the propagation delay of the bus cable. Its value is in 1/sec.

seed (G)(I4) - Represents the seed for the pseudorandom number generator. Seed is obtained by the user inputing the time of day.

session (G) - Represents the record of a network session.

session.id (G)(I) - Represents the ID number of the network session.

session.interval (G)(I4) - Represents in msec the time interval between inputs.

session.number (G)(I4) - Represents the number of characters inputed by each input.

session.quantity (G)(I4) - Represents the number of times a session is repeated in a simulation run.

session.speed (G)(I4) - Represents in bits/sec the data rate of a session's terminal.

session.variance (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

T - Is not an actual variable used in the program. T represents the time interval of the simulation clock. One T equals the time interval equating to 1/2 the round trip propagation delay of the cable.

table[x,1] (G)(I4) - Represents the ID number of a session

corresponding to the value of x.

table[x,2] (G)(I4) - Represents the number of characters per input for a sessions corresponding to the value of x.

tem - same as temp

temp (G) - Represents a record of a session's parameters for use in the simulation run.

temp.bus-busy-time (G)(I4) - Represents the total time the bus is busy because of a particular session. Time is in T intervals and is set to 0 in this program.

temp.busy-time-on-bus (G)(I4) - Represents the calculated time that the bus is busy once an input reaches the BIU. Time is in T intervals.

temp.extended-id (G)(I) - Represents the extended ID number of a session.

temp.extra-time-to-tx (G)(I4) - Represents the additional amount of waiting time for each session because the bus is busy. Time is in T intervals and is set to 0 in this program.

temp.flow-control (G)(I4) - Represents the amount of time that inputs must wait because of the BIU having trouble passing the traffic it has. Time is in T intervals and is set to 0 in this program.

temp.input-interval (G)(I4) - Represents the calculated time that elapses between each input. Time is in T intervals.

temp.input-variance (G)(I4) - Represents the calculated time deviation of the input interval. Time is in T intervals.

temp.interval-to-bus (G)(I4) - Represents the calculated time that an input takes to travel from the terminal to the BIU. Time is in T intervals.

temp.number-of-collisions (G)(I4) - Represents the number of

collisions for a particular session and is set to 0 in this program.

temp.session-id (G)(I4) - Represents the ID number of the session.

temp.time-before-next-input (G)(I4) - Represents the randomly picked time for the first input to occur. Time is in T intervals.

temp.time-to-bus (G)(I4) - Represents the needed elapse time before an input reaches the BIU. Time is in T intervals and is set to 0 in this program.

temp.total-input-time (G)(I4) - Represents the total elapse time of time between inputs. Time is in T intervals and is set to zero in this program.

temp.traffic-inputed (G)(I4) - Represents the number of inputs by a particular session. Is set to 0 by this program.

value1 (L)(R) - Represents the time the bus will be busy once a BIU is ready to transmit. Time is in real values of T intervals.

value3 (L)(R) - Represents the fraction difference between a real value of busy-time-on-bus and an integer value of busy-time-on-bus.

value4 (L)(R) - Represents the percentage of error introduced by using an integer value of busy-time-on-bus instead of a real value.

w (L)(R) - Is used for calculations involving large numbers, where intermediate calculations may exceed the allowable range of the integer4.

wild (L)(C) - Represents user's response to a question to continue.

x (G)(I) - Is used as a counter in loops.

y (L)(I) - Is used as a counter in loops.

z (L)(I) - Is used as a counter in loops.

The actual Pascal program, Parameters, begins on the next page.

```
            {* PROGRAM:PARAMETERS *}

{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: parameters                                                       }
{  FUNCTION: This program, using the network session file and the         }
{            parameters of the bus, calculates the parameters that        }
{           will be used during the simulation run.                       }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: none                                                 }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: input_data, adjust_data, display_results,           }
{                     check_bus_busy_time                                 }
{  CALLING PROCEDURES: none                                               }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


program parameters (input,output);

  const propagation = 656400000;   {ft/sec}

  type adjust = record
                  session_id:integer;
                  extended_id:integer;
                  time_to_bus:integer4;
                  interval_to_bus:integer4;
                  time_before_next_input:integer4;
                  input_interval:integer4;
                  input_variance:integer4;
                  busy_time_on_bus:integer4;
                  traffic_delay:integer4;
                  flow_control_delay:integer4;
                  number_of_collisions:integer4;
                  traffic_inputed:integer4;
                  bus_busy_time:integer4;
                  extra_time_to_tx:integer4;
                  total_input_time:integer4
                end;

       network_session = record
                           id:integer;
                           speed,number,interval,variance,
                           quantity:integer4
                         end;
```

```
            bus_setup = record
                          bus_speed:integer4;
                          bus_one_way_length:integer4;
                          bus_overhead:integer4
                        end;

  var session:network_session;
      bus:bus_setup;
      temp,tem:adjust;
      datafile:file of network_session;
      busfile:file of bus_setup;
      parameters:file of adjust;
      table:array[1..500,1..2] of integer4;
      bus_rate,seed:integer4;
      recipical_delay:real;
      bus_length,overhead,x,range:integer;
      answer:char;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: randominteger                                                    }
{  FUNCTION: To generate a pseudorandom number between 1 and 65536.       }
{  INPUTS: seed                                                           }
{  OUTPUTS: randominteger                                                 }
{  GLOBAL VARIABLES: seed                                                 }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: adjust_data                                        }
{  AUTHOR: Doug Cooper and Michael Clancy                                 }
{  HISTORY: This randominteger function is taken from the text book,      }
{           'Oh! Pascal', by Doug Cooper and Michael Clancy.              }
{*************************************************************************}


  function randominteger (var seed:integer4):integer4;

    const modulus = 65536;
          multiplier = 25173;
          increment = 13849;

    begin
      seed := ((multiplier * seed) + increment) mod modulus;
      randominteger := 1 + trunc4(modulus * (seed/modulus))
    end;


{*************************************************************************}
```

```
{  DATE: 30 Aug 84                                                          }
{  VERSION 1.0                                                              }
{  NAME: input_data                                                         }
{  FUNCTION: This procedure prompts the user for bus parameters, such       }
{            as, bus speed, bus length and bus overhead.  It also           }
{           request a seed for the random number generator.                 }
{  INPUTS: none                                                             }
{  OUTPUTS: none                                                            }
{  GLOBAL VARIABLES: bus, bus_rate, bus_length, overhead, seed,             }
{                    recipical_delay, busfile                               }
{  GLOBAL TABLES USED: none                                                 }
{  GLOBAL TABLES CHANGED: none                                              }
{  FILES READ: none                                                         }
{  FILES WRITTEN: busfile                                                   }
{  PROCEDURES CALLED: none                                                  }
{  CALLING PROCEDURES: parameters                                           }
{  AUTHOR: Capt John M. Schriml                                             }
{  HISTORY: none                                                            }
{***************************************************************************}


  procedure input_data;

    begin
      writeln;
      writeln ('Please enter data rate of the bus.<CR>');
      readln (bus_rate);
      writeln;
      write ('Please enter, in feet, the maximum round trip length ');
      writeln ('of the bus.<CR>');
      readln (bus_length);
      bus_length := bus_length div 2;
      writeln;
      writeln ('Please enter the overhead of the BIU.<CR>');
      readln (overhead);
      writeln;
      writeln ('Enter the time of day in hours and minutes.');
      writeln ('(example: for 8:45 enter 845)');
      writeln ('<CR>');
      readln (seed);
      writeln;
      recipical_delay := propagation / bus_length;
      rewrite (busfile);
      bus.bus_speed := bus_rate;
      bus.bus_one_way_length := bus_length;
      bus.bus_overhead := overhead;
      busfile^ := bus;
      put (busfile);
      close (busfile)
    end;


{***************************************************************************}
```

```
{  DATE: 30 Aug 84                                                          }
{  VERSION: 1.0                                                             }
{  NAME: adjust_data                                                        }
{  FUNCTION: Using the inputed networks sessions and the bus                }
{            parameters, this procedure calculates the actual values        }
{            that will be used during the simulation.                       }
{  INPUTS: none                                                             }
{  OUTPUTS: none                                                            }
{  GLOBAL VARIABLES: datafile, parameters, range, session, temp,            }
{                    recipical_delay, bus_rate, overhead, x                 }
{  GLOBAL TABLES USED: none                                                 }
{  GLOBAL TABLES CHANGED: table                                             }
{  FILES READ: network                                                      }
{  FILES WRITTEN: parameters                                                }
{  PROCEDURES CALLED: function randominteger                                }
{  CALLING PROCEDURES: parameters                                           }
{  AUTHOR: Capt John M. Schriml                                             }
{  HISTORY: none                                                            }
{***************************************************************************}


  procedure adjust_data;

    const modulus = 65536;

    var z,y:integer;
        w:real;

    begin
      y := 0;
      reset (datafile);
      rewrite (parameters);
      while not eof (datafile) do
        begin
          y := y + 1;
          range := y;
          session := datafile^;
          get (datafile);
          table[y,1] := session.id;
          table[y,2] := session.number;
          z := ord(session.quantity);
          for x := 1 to z do
            begin
              temp.session_id := session.id;
              temp.extended_id := x;
              w := (recipical_delay * 8 * session.number)/session.speed;
              temp.interval_to_bus := trunc4(w);
              temp.time_to_bus := 0;
              w := (session.interval * recipical_delay)/1000;
              temp.input_interval := trunc4(w);
              w := ( w * randominteger(seed)) / modulus;
              temp.time_before_next_input := trunc4(w);
              w := (session.variance * recipical_delay) / 1000;
```

```
            temp.input_variance := trunc4(w);
            w := (((session.number * 8) + overhead) * recipical_delay)
                        / bus_rate;
            temp.busy_time_on_bus := trunc4(w);
            temp.traffic_delay := 0;
            temp.flow_control_delay := 0;
            temp.number_of_collisions := 0;
            temp.traffic_inputed := 0;
            temp.bus_busy_time := 0;
            temp.extra_time_to_tx := 0;
            temp.total_input_time := 0;
            parameters^ := temp;
            put (parameters)
          end
      end;
    close (datafile);
    close (parameters)
  end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: display_results                                                  }
{  FUNCTION: Upon user's request, the procedure displays a copy of        }
{            the data that will be used in the simulation.                }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: parameters, tem                                      }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: parameters                                                 }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: parameters                                         }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure display_results;

    var wild:char;

    begin
      writeln;
      write ('                    TIME       BUS        NEXT');
      writeln ('     INPUT        INPUT       BUS');
      write (' ID   EXT ID     TO BUS    INTERVAL    INPUT');
      writeln ('    INTERVAL   VARIANCE   BUSY TIME');
      writeln;
      writeln;
      reset (parameters);
```

```
        end;
      if (bus_event <= input_event) and (bus_event <> 0)
        then begin
          next_event := bus_event;
          event_flag := 'b'
        end
        else if (bus_event < input_event) and (bus_event = 0)
          then begin
            next_event := input_event;
            event_flag := 'i'
          end
          else if (input_event < bus_event) and (input_event <> 0)
            then begin
              next_event := input_event;
              event_flag := 'i'
            end
            else begin
              next_event := bus_event;
              event_flag := 'b'
            end
  end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME:  prepare_for_event                                               }
{  FUNCTION: Having found the next_event, this procedure reduces all      }
{            appropriate table values by next_event - 1. The next         }
{            event can then be identified by the value 1.  Also how       }
{            many and what sessions have a number 1 is determined.        }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: waiting, whose_waiting, range, bus_event,            }
{                    next_event, table                                    }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: table                                           }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: simulate                                           }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure prepare_for_event;

    var temp1,temp2,temp3:integer4;
        z:integer;

    begin
      waiting := 0;
```

```
{  INPUTS: none                                                              }
{  OUTPUTS: none                                                             }
{  GLOBAL VARIABLES: waiting, whose_waiting, bus_event, input_event,         }
{                    next_event, table, event_flag                           }
{  GLOBAL TABLES USED: table                                                 }
{  GLOBAL TABLES CHANGED: none                                               }
{  FILES READ: none                                                          }
{  FILES WRITTEN: none                                                       }
{  PROCEDURES CALLED: none                                                   }
{  CALLING PROCEDURES: simulate                                              }
{  AUTHOR: Capt John M. Schriml                                              }
{  HISTORY: none                                                             }
{*****************************************************************************}


  procedure find_next_event;

    var temp,tem:integer4;
        y,z:integer;

    begin
      waiting := 0;
      whose_waiting := 0;
      bus_event := 0;
      for z := 1 to range do
        begin
          if table[z,1] = 1
            then whose_waiting := z;
          if table[z,1] < 1000
            then limiter := trunc(table[z,1])
            else limiter := 1000;
          if limiter in limitset
            then waiting := waiting + 1;
          if table[z,1] <> 0
            then begin
              tem := table[z,1];
              if bus_event = 0
                then bus_event := tem
                else if tem < bus_event
                  then bus_event := tem
            end
        end;
       input_event := 0;
       for y := 1 to range do
         begin
           if table[y,3] <> 0
             then begin
               temp := table[y,3];
               if input_event = 0
                 then input_event := temp
                 else if temp < input_event
                   then input_event := temp
             end
```

```
{  OUTPUTS: none                                                              }
{  GLOBAL VARIABLES: table, datafile, range                                   }
{  GLOBAL TABLES USED: table                                                  }
{  GLOBAL TABLES CHANGED: table                                               }
{  FILES READ: parameters                                                     }
{  FILES WRITTEN: none                                                        }
{  PROCEDURES CALLED: none                                                    }
{  CALLING PROCEDURES: simulate                                               }
{  AUTHOR: Capt John M. Schriml                                               }
{  HISTORY: none                                                              }
{*****************************************************************************}


  procedure get_data;

    begin
      reset (datafile);
      x := 0;
      while not eof (datafile) do
        begin
          comp := datafile^;
          get (datafile);
          x := x + 1;
          table[x,1] := comp.time_to_bus;
          table[x,2] := comp.interval_to_bus;
          table[x,3] := comp.time_before_next_input;
          table[x,4] := comp.input_interval;
          table[x,5] := comp.busy_time_on_bus;
          table[x,6] := comp.traffic_delay;
          table[x,7] := comp.flow_control_delay;
          table[x,8] := comp.number_of_collisions;
          table[x,9] := comp.inputed_traffic;
          table[x,10] := comp.bus_busy_time;
          table[x,11] := comp.extra_time_to_tx;
          table[x,12] := 0;
          table[x,13] := 0;
          table[x,14] := comp.input_variance;
          table[x,15] := 0;
        end;
      range := x;
      close (datafile)
    end;


{*****************************************************************************}
{  DATE: 30 AUG 84                                                            }
{  VERSION: 1.0                                                               }
{  NAME: find_next_event                                                      }
{  FUNCTION: To scan the matrix table and to determine what event will        }
{            occur next. Also to determine if the event is a bus event        }
{            or an input event. If the next event is a one, the               }
{            procedure determines where the event is located in the           }
{            matrix table.                                                    }
```

```
  var table:array[1..500,1..15] of integer4;
      event_flag,screen,answer:char;
      waiting,whose_waiting,x,range,limiter,N:integer;
      percentage,persistant:integer;
      seed:integer4;
      sclock,tclock,clock,next_event,bus_event,input_event:integer4;
      comp,sim:data;
      clc:clck;
      clockfile:file of clck;
      datafile,simfile:file of data;
      limitset:limit;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: randominteger                                                    }
{  FUNCTION: To generate a pseudorandom number between 1 and 65536.       }
{  INPUTS: seed                                                           }
{  OUTPUTS: randominteger                                                 }
{  GLOBAL VARIABLES: seed                                                 }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGES: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: collision, adj_input, BB, CC                       }
{  AUTHOR: Doug Cooper and Michael Clancy                                 }
{  HISTORY: This randominteger function is taken from the text book,      }
{           'Oh! Pascal', by Doug Cooper and Michael Clancy.              }
{*************************************************************************}


  function randominteger (var seed:integer4):integer4;

    const modulus = 65536;
          multiplier = 25173;
          increment = 13849;

    begin
      seed := ((multiplier * seed) + increment) mod modulus;
      randominteger := 1 + trunc4(modulus * (seed/modulus))
    end;


{*************************************************************************}
{  DATE: 30 AUG 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: get_data                                                         }
{  FUNCTION: This procedure loads the parameters file into the matrix     }
{            table in preparation for the simulation.                     }
{  INPUTS: none                                                           }
```

```
                {* PROGRAM:SIMULATE *}

{*************************************************************************}
{  DATE: 10 Sep 84                                                        }
{  VERSION: 1.1                                                           }
{  NAME: simulate                                                         }
{  FUNCTION: Based on the data received from the parameters file, this    }
{            program will simulate a local area network.                  }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: range, answer, seed, tclock, sclock, clock, clc,      }
{                    screen, next_event, event_flag, table, clockfile     }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: table                                           }
{  FILES READ: none                                                       }
{  FILES WRITTEN: clock                                                   }
{  PROCEDURES CALLED: get_data, display, find_next_event, adj_bus,        }
{                     adj_input, prepare_for_event, file_results, test    }
{  CALLING PROCEDURES: none                                               }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


program simulate (input,output);

  type data = record
                session_id:integer;
                extended_id:integer;
                time_to_bus:integer4;
                interval_to_bus:integer4;
                time_before_next_input:integer4;
                input_interval:integer4;
                input_variance:integer4;
                busy_time_on_bus:integer4;
                traffic_delay:integer4;
                flow_control_delay:integer4;
                number_of_collisions:integer4;
                inputed_traffic:integer4;
                bus_busy_time:integer4;
                extra_time_to_tx:integer4;
                total_input_time:integer4
              end;

       clck = record
                T_click,S_click:integer4;
                delay,persistant,percentage:integer
              end;

      delay = 1..250;
      limit = set of delay;
```

more than one session waiting to use the bus, this variable is not used, but if there is only one session waiting, this variable will indicate which row the session is in.

x (G)(I) - Is used for loops.

y (L)(I) - Is used for loops.

z (L)(I) - Is used for loops.

z (L)(R) - Represents the number of times the T interval can be divide into the actual delay of the cable.

The actual Pascal program, Simulate, begins on the next page.

actually busy. Time is in T intervals.

table[x,11] (G)(I4) - This column sums up the extra waiting time due to collisions. Time is in T intervals.

table[x,12] (G)(I4) - This column stores the local parameter used with the exponential binary backoff.

table[x,13] (G)(I4) - This column temporarily stores the extra time a session's traffic must wait because of collisions. As the simulation clock counts, portions of table[x,13] are removed and added to table[x,11]. Time is in T intervals.

table[x,14] (G)(I4) - Represents the calculated time for the +/- deviation of the interval between inputs. Time is in T intervals.

table[x,15] (G)(I4) - This column sums up the actual time between inputs. Time is in T intervals.

tclock (G)(I4) - Represents the amount time the simulation is ran while collecting data. Time is in T intervals.

tem (L)(I4) - Represents a temporary storage location for the next bus-event.

temp (L)(I4) - Represents a temporary storage location for the next input-event.

temp1 (L)(I4) - Represents an intermediate calculation.

temp2 (L)(I4) - Represents an intermediate calculation.

temp3 (L)(I4) - Represents an intermediate calculation.

value1 (L)(I4) - Represents an intermediate calculation.

waiting (G)(I) - Represents the number of sessions waiting to use the bus.

whose-waiting (G)(I4) - Represents the last session in the matrix column, table[x,1], that wants to use the busy. Normally if there are

session's traffic must wait due to a collision.

sim (G) - same as comp.

simfile (G) - Represents a file of records containing the session's parameters and performance data.

T - Is not an actual variable used in the program. T is the time interval of the simulation clock. T is equal to the time equating to 1/2 the round trip propagation delay of the bus cable.

table[x,1] (G)(I4) - Represents the amount of elapse time before an input reaches the BIU. Time is in T intervals.

table[x,2] (G)(I4) - Represents the calculated amount of time for an input to travel from the terminal to BIU. Time is in T intervals.

table[x,3] (G)(I4) - Represents the amount of elapse time before an input is made. Time is in T intervals.

table[x,4] (G)(I4) - Represents the calculated amount time between inputs. Time is in T intervals.

table[x,5] (G)(I4) - Represents the calculated time that the bus is busy once a BIU begins to transmit. Time is in T intervals.

table[x,6] (G)(I4) - This column sums up all delay time due to the bus being busy. Time is in T intervals.

table[x,7] (G)(I4) - This column sums up all input delay time due to flow control being employed. Time is in T intervals.

table[x,8] (G)(I4) - This column sums up all collisions. Only table[1,8] is used for counting collisions. All other values of table[x,8] are 0 and used as dummy variables to maintain the same record type.

table[x,9] (G)(I4) - This column sums up the number of inputs.

table[x,10] (G)(I4) - This column sums up the time the bus is

cable.

limiter (G)(I) - Represents the value of table[x,1] in integer form, if table[x,1] is less than 1000.

limitset (G)(I) - Represents the interval between the next tick of the clock and the next tick of the clock plus the round trip delay of the bus cable.

N (G)(I) - Represents the time interval equal to one tick of the clock plus the round trip delay of the bus cable. Time is in T intervals.

next-event (G)(I4) - Represents the smallest elapse time between bus-event and input-event. Which ever event time is smaller, will become the next-event.

percentage (G)(I) - Represents the amount of persistant when simulating p-persistant.

persistant (G)(I) - Represents the value indicating either 1- or p-persistant is being simulated.

range (G)(I) - Represents the number of sessions loaded for simulation. Since the number of sessions are unknown prior to simulation, the range value is determined when loading data. Range is then used for loops and to limit the size of the table matrix.

sclock (G)(I4) - Represents the amount of simulation time used for collecting data. Time is in T intervals.

screen (G)(C) - Represents the user's response to a question asking if the user wishes to see a display of the actual simulation run.

seed (G)(I4) - Represents the seed necessary for the random number generator. The seed is obtained by asking the user to enter the time of day.

set-back (L)(R) - Represents the amount of additional time a

collisions for a particular session.

comp.session-id (G)(I) - Represents the ID number of the session.

comp.time-before-next-input (G)(I4) - Represents the amount time that must elapse before the next input is made for a particular session. Time is in T intervals.

comp.time-to-bus (G)(I4) - Represents the amount of time that must elapse before inputed traffic reaches the BIU. Time is in T intervals

comp.total-input-time (G)(I4) - Represents the total amount of elaspe time between inputs.

comp.traffic-delay (G)(I4) - Represents the total amount of time a particular session had to wait because the bus was busy.

count (L)(I4) - Represents a random number between 1 and 65536.

count1 (L)(I4) - Represents a random number between 1 and 65536.

count2 (L)(I4) - Represents a random number between 1 and 65536.

datafile (G) - Represents a file of records containing the perforamce parameters of each session.

delay (L)(R) - Represents in nano seconds the actual known delay of the bus cable.

event-flag (G)(C) - Represents the fact that the next event is either a bus event or input event.

fact (L)(R) - Represents the randomly picked amount of time between the +/- deviation of the input interval. The value is in real T intervals.

factor (L)I4) - Represents the integer4 value of fact.

input-event (G)(I4) - Represents the smallest amount of elaspe time before an input is made. Time is in T intervasl.

length (L)(R) - Represents in feet the one-way-length of the bus

persistant of the bus being simulated. This file is used later by the program Evaluate.

comp (G) - Represents a record of a session's parameters.

comp.bus-busy-time (G)(I4) - Represents the total time the bus is busy because of a particular session. Time is in T intervals.

comp.busy-time-on-bus (G)(I4) - Represents the calculated time the bus is busy once an input reaches the BIU for a particular session. Time is in T intervals.

comp.extended-id (G)(I) - Represents the extended ID of a session.

comp.extra-time-to-tx (G)(I4) - Represents the total amount of extra time to transmit because of collisions for a particular session. Time is in T intervals.

comp.flow-control-delay (G)(I4) - Represents the total amount of time inputs for a particular session were heldup because of the BIU having trouble passing the traffic it already has. Time is in T intervals.

comp.input-interval (G)(I4) - Represents the calculated time that elapses between each input for a particular session. Time is in T intervals.

comp.inputed-traffic (G)(I4) - Represents the total number of inputs for a particular session.

comp.input-variance (G)(I4) - Represents the maximum +/- time deviation of the input interval. Time is in T intervals.

comp.interval-to-bus (G)(I4) - Represents the calculated time that it takes an input to travel from the terminal to BIU. Time is in T intervals.

comp.number-of-collisions (G)(I4) - Represents the number of

PART B6: Pascal Program Simulate

The variables and their purpose, as used with the program Simulate, are as follows:

VARIABLES:

b (L)(I) - Represents the row (in matrix table) of the session that is ready to transmit on the bus.

answer (G)(C) - Represents user's response to a question.

bus-event (G)(I4) - Represents the smallest amount of elapse time before an input reaches the bus.

clc.delay (G)(I) - Represents the actual known delay of the bus cable. Value is in nano seconds.

clc.percentage (G)(I) - Represents the percenatge of persistant, if p-persistant is being simulated.

clc.persistant (G)(I) - Represents the type of persistant being simulated.

clc.S-click (G)(I4) - Represents the amount time that the simulation ran while collecting performance data. Time is in T intervals.

clc.T-click (G)(i4) - Represents the amount of transition time that the simulation ran prior to collecting performance data. Time is in T intervals.

clock (G)(I4) - Represents the clock time of the simulation. Time is in T intervals.

clockfile (G) - Represents a file which contains the transition and simulation clock times. Also it contains the delay, percentage, and

```
            begin
              writeln;
              write ('**NOTE**  The error in calculating the bus busy');
              writeln (' time for session ID #',table[y,1]:2,' is');
              write (trunc4(value4):2,' %.  The error is high because');
              writeln (' the bus busy time is approaching ');
              write ('the basic unit of time for the simulator, ');
              writeln ('one way propagation delay.  One or ');
              write ('more of the following factors may be ');
              writeln ('contributing to the high error.');
              writeln;
              writeln ('     -high bus bite rate');
              writeln ('     -long length of the bus cable');
              writeln ('     -low overhead');
              writeln ('     -low # of characters per input');
              writeln;
              write ('Due to the high error, the results of the bus');
              writeln (' busy time, indicated');
              write ('during the data evaluation stage, should be ');
              writeln ('ignored.');
              writeln ('Enter <CR> to continue');
              read (wild);
              writeln
            end
        end
    end;

begin
  assign (datafile,'network');
  assign (busfile,'bus');
  assign (parameters,'net_data');
  input_data;
  adjust_data;
  writeln ('Do you wish to see a display of the data? (Y/N)');
  readln (answer);
  if (answer ='y') or (answer ='Y')
    then display_results;
  check_bus_busy_time
end.
```

```
      while not eof (parameters) do
        begin
          tem := parameters^;
          get (parameters);
          write (tem.session_id:3,tem.extended_id:7);
          write (tem.time_to_bus:12,tem.interval_to_bus:12);
          write (tem.time_before_next_input:10);
          write (tem.input_interval:11,tem.input_variance:11);
          writeln (tem.busy_time_on_bus:11)
        end;
      close (parameters);
      writeln;
      writeln ('Enter <CR> to continue.');
      read (wild);
      writeln
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: check_bus_busy_time                                              }
{  FUNCTION: This procedure checks the error in calculating the bus       }
{            busy time for each session.  If the error is greater         }
{            than 1 %, it advises the user.                               }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: bus_rate, overhead, recipical_delay                  }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: parameters                                         }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure check_bus_busy_time;

    var y:integer;
        wild:char;
        value1,value3,value4:real;

    begin
      for y := 1 to range do
        begin
          value1 := (((table[y,2] * float4(8)) + overhead) *
                      recipical_delay) / bus_rate;
          value3 := value1 - trunc4(value1);
          value4 := (value3/value1) * 100;
          if value4 > 1 then
```

```
        whose_waiting := 0;
        for z := 1 to range do
          begin
            if bus_event <> 0 then
              begin
                temp1 := table[z,1] - (next_event - 1);
                if temp1 > 0
                  then table[z,1] := temp1
              end;
            if table[z,1] = 1
              then whose_waiting := z;
            if table[z,1] < 1000
             then limiter := trunc(table[z,1])
             else limiter := 1000;
            if limiter in limitset
              then waiting := waiting + 1;
            temp2 := table[z,3] - (next_event -1);
            if (temp2 > 0) then
              begin
                table[z,3] := temp2;
                table[z,15] := table[z,15] + next_event - 1
              end
              else table[z,7] := table[z,7] - temp2;
            if table[z,13] <> 0 then
              begin
                temp3 := table[z,13] - (next_event - 1);
                if temp3 >= 0 then
                  begin
                    table[z,11] := table[z,11] + (next_event - 1);
                    table[z,13] := temp3
                  end
                  else begin
                    table[z,11] := table[z,11] + table[z,13];
                    table[z,13] := 0
                  end
              end
          end
      end;

{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: adj_input                                                        }
{  FUNCTION: The procedure find_next_event has determined that the        }
{            next event is an input event.  This procedure adjust         }
{            all appropriate table values based on an input event.        }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: table, range, seed                                   }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: table                                           }
{  FILES READ: none                                                       }
```

```
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: randominteger                                       }
{  CALLING PROCEDURES: simulate                                           }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure adj_input;

    var z:integer;
        count1,count2,factor:integer4;
        fact:real;

    begin
      for z := 1 to range do
        begin
          if (table[z,3] = 1) and (table[z,1] = 0) then
            begin
              count1 := randominteger(seed);
              count2 := randominteger(seed);
              fact := (float4(count1) * table[z,14]) / 65536;
              factor := trunc4(fact);
              if count2 > 32768
                then factor := 0 - factor;
              table[z,3] := table[z,4] + 1 + factor;
              table[z,9] := table[z,9] + 1;
              table[z,1] := table[z,2] + 1
            end
            else if (table[z,3] = 1) and (table[z,1] > 0 )
              then table[z,3] := 0
        end
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: AA                                                               }
{  FUNCTION: Having determined that there is no collision and that the    }
{            next event is a bus_event, this procedure makes the          }
{            necessary table adjustments for a session with a 1 sitting   }
{            on the bus and the next input value greater than the bus     }
{            busy time.                                                   }
{  INPUTS: b,z                                                            }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: table, clock                                         }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: table                                           }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: adj_bus                                            }
```

```
{  AUTHOR: Capt John M. Schriml                                          }
{  HISTORY: none                                                        }
{************************************************************************}


  procedure AA (z,b:integer);

    begin
      table[z,1] := 0;
      table[z,3] := table[z,3] - (table[b,5] + N);
      table[z,15] := table[z,15] + table[b,5] + N;
      table[z,10] := table[z,10] + table[b,5];
      clock := clock - (table[b,5] + N)
    end;


{************************************************************************}
{  DATE: 30 Aug 84                                                      }
{  VERSION: 1.0                                                         }
{  NAME: BB                                                             }
{  FUNCTION:  With no collision and a bus input, this procedure makes   }
{             the appropriate changes for a session with a 1 sitting    }
{             on the bus and the next input value less than the bus     }
{             busy time.                                                }
{  INPUTS: z,b                                                          }
{  OUTPUTS: none                                                        }
{  GLOBAL VARIABLES: table, seed, clock                                 }
{  GLOBAL TABLES USED: table                                            }
{  GLOBAL TABLES CHANGED: table                                         }
{  FILES READ: none                                                     }
{  FILES WRITTEN: none                                                  }
{  PROCEDURES CALLED: randominteger                                     }
{  CALLING PROCEDURES: adj_bus                                          }
{  AUTHOR: Capt John M. Schriml                                         }
{  HISTORY: none                                                        }
{************************************************************************}


  procedure BB (z,b:integer);

    var temp1,temp2,temp3,factor,count1,count2:integer4;
        fact:real;

    begin
      temp1 := (table[b,5] + N) - table[z,3];
      table[z,15] := table[z,15] + table[z,3];
      if table[z,3] < 1 + 2 * (N - 1) then
        begin
          table[z,7] := table[z,7] +1+ (2 * (N - 1)) - table[z,3];
          temp2 := table[z,2] -(table[b,5] - (N -1))
        end
        else temp2 := table[z,2] - temp1;
      count1 := randominteger(seed);
```

```
      count2 := randominteger(seed);
      fact := (table[z,14] * float4(count1)) / 65536;
      factor := trunc4(fact);
      if count2 > 37768
        then factor := 0 - factor;
      if table[z,3] < 1 + 2 * (N - 1)
        then temp3 := (table[z,4] + factor) - (table[b,5] - (N-1))
        else temp3 := (table[z,4] + factor) - temp1;
      if temp3 > 0 then
        begin
          if table[z,3] < 1 + 2 * (N - 1)
            then table[z,15] := table[z,15] + (table[b,5] - (N - 1))
            else table[z,15] := table[z,15] + temp1;
          table[z,3] := temp3
        end
        else begin
          table[z,3] := 0;
          table[z,15] := table[z,15] + (table[z,4] + factor);
          table[z,7] := table[z,7] - temp3
        end;
      table[z,9] := table[z,9] + 1;
      table[z,10] := table[z,10] + table[b,5];
      clock := clock - (table[b,5] + N);
      if (temp2 > 0)
        then table[z,1] := temp2
        else begin
          table[z,1] := 1;
          table[z,6] := table[z,6] - temp2
        end
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: CC                                                               }
{  FUNCTION: With no collision and a bus input, this procedure makes      }
{            the appropriate changes for a session with 0 on the bus      }
{            and the next input event less than the bus busy time.        }
{  INPUTS: z,b                                                            }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: table, seed                                          }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: table                                           }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: randominteger                                       }
{  CALLING PROCEDURES: adj_bus                                            }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}
```

```
  procedure CC (z,b:integer);

    var temp1,temp2,factor,count1,count2:integer4;
        fact:real;

    begin
      temp1 := (table[b,5] + N) - table[z,3];
      temp2 := table[z,2] - temp1;
      table[z,15] := table[z,15] + table[b,5] + N;
      count1 := randominteger(seed);
      count2 := randominteger(seed);
      fact := (table[z,14] * float4(count1)) / 65536;
      factor := trunc4(fact);
      if count2 > 32768
        then factor := 0 - factor;
      table[z,3] := (table[z,4] + factor) - temp1;
      table[z,9] := table[z,9] + 1;
      if (temp2 > 0)
        then table[z,1] := temp2
        else begin
          table[z,1] := 1;
          table[z,6] := table[z,6] - temp2
        end
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: DD                                                               }
{  FUNCTION: With no collision and a bus event, this procedure makes      }
{            the appropriate changes for a session with 0 on the bus      }
{            and the next input event is greater than the bus busy        }
{            time.                                                        }
{  INPUTS: z,b                                                            }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: table                                                }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: table                                           }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: adj_bus                                            }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure DD (z,b:integer);

    begin
      table[z,15] := table[z,15] + table[b,5] + N;
      table[z,3] := table[z,3] - (table[b,5] + N)
```

```
    end;

{************************************************************************}
{  DATE: 30 Aug 84                                                       }
{  VERSION: 1.0                                                          }
{  NAME: EE                                                              }
{  FUNCTION: With no collision and a bus event, this procedure makes     }
{            the appropriate changes for a session with bus time less    }
{            than the bus busy time and the input time equal to 0.       }
{  INPUTS: z,b                                                           }
{  OUTPUTS: none                                                         }
{  GLOABAL VARIABLES: table                                              }
{  GLOBAL TABLES USED: table                                             }
{  GLOBAL TABLES CHANGED: table                                          }
{  FILES READ: none                                                      }
{  FILES WRITTEN: none                                                   }
{  PROCEDURES CALLED: none                                               }
{  CALLING PROCEDURES                                                    }
{  AUTHOR: Capt John M. Schriml                                          }
{  HISTORY: none                                                         }
{************************************************************************}


  procedure EE (z,b:integer);

    var temp1:integer4;

    begin
      temp1 := (table[b,5] + N) - table[z,1];
      table[z,1] := 1;
      table[z,6] := table[z,6] + temp1;
      table[z,7] := table[z,7] + (table[b,5] + N) - 1
    end;


{************************************************************************}
{  DATE: 30 Aug 84                                                       }
{  VERSION: 1.0                                                          }
{  NAME: FF                                                              }
{  FUNCTION: With no collision and a bus input, this procedure makes     }
{            the appropriate changes for a session with bus time         }
{            greater than the bus busy time and the next input time      }
{            equals 0.                                                   }
{  INPUTS: z,b                                                           }
{  OUTPUTS: none                                                         }
{  GLOBAL VARIABLES: table                                               }
{  GLOBAL TABLES USED: table                                             }
{  GLOBAL TABLES CHANGED: table                                          }
{  FILES READ: none                                                      }
{  FILES WRITTEN: none                                                   }
{  PROCEDURES CALLED: none                                               }
{  CALLING PROCEDURES: adj_bus                                           }
```

```
{  AUTHOR: Capt John M. Schriml                                              }
{  HISTORY: none                                                             }
{*****************************************************************************}


  procedure FF (z,b:integer);

    begin
      table[z,1] := table[z,1] - (table[b,5] + N);
      table[z,7] := table[z,7] + (table[b,5] + N)
    end;


{*****************************************************************************}
{  DATE: 30 Aug 84                                                           }
{  VERSION: 1.0                                                              }
{  NAME: GG                                                                  }
{  FUNCTION: With no collision and a bus input, this procedure makes         }
{            the appropriate changes for a session bus time and input        }
{            time greater than the bus busy time.                            }
{  INPUTS: z,b                                                               }
{  OUTPUTS: none                                                             }
{  GLOBAL VARIABLES: table                                                   }
{  GLOBAL TABLES USED: table                                                 }
{  GLOBAL TABLES CHANGED: table                                              }
{  FILES READ: none                                                          }
{  FILES WRITTEN: none                                                       }
{  PROCEDURES CALLED: none                                                   }
{  CALLING PROCEDURES: adj_bus                                               }
{  AUTHOR: Capt John M. Schriml                                              }
{  HISTORY: none                                                             }
{*****************************************************************************}


  procedure GG (z,b:integer);

    begin
      table[z,15] := table[z,15] + table[b,5] + N;
      table[z,1] := table[z,1] - (table[b,5] + N);
      table[z,3] := table[z,3] - (table[b,5] + N)
    end;


{*****************************************************************************}
{  DATE: 30 Aug 84                                                           }
{  VERSION: 1.0                                                              }
{  NAME: HH                                                                  }
{  FUNCTION: With no collision and a bus input, this procedure makes         }
{            the appropriate changes for a session with bus time and         }
{            input time less than the bus busy time.                         }
{  INPUTS: z,b                                                               }
{  OUTPUTS: none                                                             }
{  GLOBAL VARIABLES: table                                                   }
```

```
{  GLOBAL TABLES USED: table                                                  }
{  GLOBAL TABLES CHANGED: table                                               }
{  FILES READ: none                                                           }
{  FILES WRITTEN: none                                                        }
{  PROCEDURES CALLED: none                                                    }
{  CALLING PROCEDURES: adj_bus                                                }
{  AUTHOR: Capt John M. Schriml                                               }
{  HISTORY: none                                                              }
{*************************************************************************}

  procedure HH (z,b:integer);

    var temp1,temp2:integer4;

    begin
      temp1 := (table[b,5] + N) - table[z,1];
      temp2 := (table[b,5] + N) - table[z,3];
      table[z,15] := table[z,15] + table[z,3];
      table[z,1] := 1;
      table[z,3] := 0;
      table[z,7] := table[z,7] + temp2 - 1;
      table[z,6] := table[z,6] + temp1
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                            }
{  VERSION: 1.0                                                               }
{  NAME: II                                                                   }
{  FUNCTION: With no collision and a bus input, this procedure makes          }
{            the appropriate changes for a session with bus time              }
{            greater than bus busy time and input time less than bus          }
{            busy time.                                                       }
{  INPUTS: z,b                                                                }
{  OUTPUTS: none                                                              }
{  GLOBAL VARIABLES: table                                                    }
{  GLOBAL TABLES USED: table                                                  }
{  GLOBAL TABLES CHANGED: table                                               }
{  FILES READ: none                                                           }
{  FILES WRITTEN: none                                                        }
{  PROCEDURES CALLED: none                                                    }
{  CALLED PROCEDURES: adj_bus                                                 }
{  AUTHOR: Capt John M. Schriml                                               }
{  HISTORY: none                                                              }
{*************************************************************************}

  procedure II (z,b:integer);

    var temp1:integer4;

    begin
```

```
      temp1 := (table[b,5] + N) - table[z,3];
      table[z,1] := table[z,1] - (table[b,5] + N);
      table[z,15] := table[z,15] + table[z,3];
      table[z,3] := 0;
      table[z,7] := table[z,7] + temp1
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: JJ                                                               }
{  FUNCTION: With no collision and a bus input, this procedure makes      }
{            the appropriate changes for a session with bus time less     }
{            than bus busy time and input time greater than bus busy      }
{            time.                                                        }
{  INPUTS: z,b                                                            }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: table                                                }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: table                                           }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: adj_bus                                            }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure JJ (z,b:integer);

    var temp1:integer4;

    begin
      temp1 := (table[b,5] + N) - table[z,1];
      table[z,1] := 1;
      table[z,15] := table[z,15] + table[b,5] + N;
      table[z,3] := table[z,3] - (table[b,5] + N);
      table[z,6] := table[z,6] + temp1
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: collision                                                        }
{  FUNCTION: Having determined a collision will occur, this procedure     }
{            generates the time which each session is set back.           }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: range, table, seed                                   }
{  GLOBAL TABLES USED: table                                              }
```

```
{  GLOBAL TABLES CHANGED: table                                              }
{  FILES READ: none                                                          }
{  FILES WRITTEN: none                                                       }
{  PROCEDURES CALLED: randominteger                                          }
{  CALLING PROCEDURES: adj_bus                                               }
{  AUTHOR: Capt John M. Schriml                                              }
{  HISTORY: none                                                             }
{*************************************************************************}


  procedure collision;

    var z:integer;
        count:integer4;
        set_back:real;

    begin
      writeln ('COLLISION');
      table[1,8] := table[1,8] + 1;
      for z := 1 to range do
        begin
          if table[z,1] < 1000
            then limiter := trunc(table[z,1])
            else limiter := 1000;
          if limiter in limitset
            then begin
              count := randominteger(seed);
              if table[z,12] = 0
                then table[z,12] := 2
                else table[z,12] := table[z,12] * 2;
              set_back := table[z,12] * ((table[z,5] * float4(count)) /
                            65536);
              table[z,1] := table[z,1] + (N-1) div 2 + trunc4(set_back);
              if table[z,3] < 1 + 2 * (N - 1)
                then table[z,7] := table[z,7] + 1 + 2 * (N-1)
                          - table[z,3];
              table[z,11] := table[z,11] + (N - 1) div 2;
              table[z,13] := trunc4(set_back)
            end
        end
    end;


{*************************************************************************}
{  DATE: 17 Sep 84                                                           }
{  VERSION: 1.0                                                              }
{  NAME: adj_further                                                         }
{  Function: This procedure simulates the traffic's decision to              }
{            transmit or not transmit when using p-persistant.               }
{  INPUT: none                                                               }
{  OUTPUTS: none                                                             }
{  GLOBAL VARIABLES: seed, table, percentage                                 }
{  GLOBAL TABLES USED: table                                                 }
```

```
{  GLOBAL TABLES CHANGED: table                                              }
{  FILES READ: none                                                          }
{  FILES WRITTEN: none                                                       }
{  PROCEDURES CALLED: none                                                   }
{  CALLING PROCEDURES: simulate                                              }
{  AUTHOR: Capt John M. Schriml                                              }
{  HISTORY: none                                                             }
{*****************************************************************************}


  procedure adj_further;

    var z:integer;
        value1,count:integer4;

    begin
      waiting := 0;
      for z := 1 to range do
        begin
          if table[z,1] <1000
            then limiter := trunc(table[z,1])
            else limiter := 1000;
          if limiter in limitset then
            begin
              count := randominteger(seed);
              value1 := (percentage * 65536) div 100;
              if count > value1 then
                begin
                  table[z,1] := table[z,1] + N;
                  table[z,6] := table[z,6] + N
                end;
              limiter := trunc(table[z,1]);
              if limiter in limitset
                then waiting := waiting + 1
            end
        end
    end;


{*****************************************************************************}
{  DATE: 30 Aug 84                                                           }
{  VERSION: 1.0                                                              }
{  NAME: storage                                                             }
{  FUNCTION: To add portions of sessions set_back time to the data           }
{            collecting part of the matrix table.                            }
{  INPUTS: z,b                                                               }
{  OUTPUTS: none                                                             }
{  GLOBAL VARIABLES: table                                                   }
{  GLOBAL TABLES USED: table                                                 }
{  GLOBAL TABLES CHANGED: table                                              }
{  FILES READ: none                                                          }
{  FILES WRITTEN: none                                                       }
{  PROCEDURES CALLED: none                                                   }
```

```
{  CALLING PROCEDURES: adj_bus                                              }
{  AUTHOR: Capt John M. Schriml                                             }
{  HISTORY: none                                                            }
{***************************************************************************}


  procedure storage (z,b:integer);

    var temp:integer4;

    begin
      if table[z,13] <> 0
      then begin
        temp := table[z,13] - (table[b,5] + N);
        if temp > 0 then
          begin
            table[z,11] := table[z,11] + (table[b,5] + N);
            table[z,13] := temp
          end
          else begin
            table[z,11] := table[z,11] + table[z,13];
            table[z,13] := 0
          end
       end
    end;


{***************************************************************************}
{  DATE: 30 Aug 84                                                          }
{  VERSION: 1.0                                                             }
{  NAME: adj_bus                                                            }
{  FUNCTION: Having determined that an event will occur, the procedure      }
{            checks the status of each session's bus input time and         }
{            input time, and calls the appropriate procedure to adjust      }
{            each session based on its status.                              }
{  INPUTS: none                                                             }
{  OUTPUTS: none                                                            }
{  GLOBAL VARIABLES: table, waiting, whose_waiting, range                   }
{  GLOBAL TABLES USED: table                                                }
{  GLOBAL TABLES CHANGED: none                                              }
{  FILES READ: none                                                         }
{  FILES WRITTEN: none                                                      }
{  PROCEDURES CALLED: AA - JJ, collision, storage                           }
{  CALLING PROCEDURE: simulate                                              }
{  AUTHOR: Capt John M. Schriml                                             }
{  HISTORY: none                                                            }
{***************************************************************************}


  procedure adj_bus;

    var z,b:integer;
```

```
begin
  if waiting = 1 then
    begin
      b := whose_waiting;
      for z := 1 to range do
       begin
        if (z = b) and (table[z,3] > table[b,5] + 1 + 2 * (N - 1))
         then AA (z,b)
         else if (z=b) and (table[z,3] <= table[b,5] + 1 + 2*(N-1))
          then BB (z,b)
          else if (table[z,1] = 0) and (table[z,3] <= table[b,5] +N)
           then CC (z,b)
           else if (table[z,1] = 0) and (table[z,3] > table[b,5] +N)
            then DD (z,b)
            else if (table[z,1] <= table[b,5] + N) and
                                (table[z,3] = 0) and (table[z,1] <> 0)
             then EE (z,b)
             else if (table[z,1] > table[b,5] + N) and
                                  (table[z,3] = 0)
              then FF (z,b)
              else if (table[z,1] > table[b,5] + N) and
                                   (table[z,3] > table[b,5] + N)
               then GG (z,b)
               else if (table[z,1] <= table[b,5] + N) and
                                   (table[z,3] <= table[b,5] + N)
                then HH (z,b)
                else if (table[z,1] > table[b,5] + N) and
                                    (table[z,3] <= table[b,5] + N)
                 then II (z,b)
                 else if (table[z,1] <= table[b,5] + N) and
                                     (table[z,3] > table[b,5] + N)
                   then JJ (z,b);
           table[z,12] := 0;
           storage (z,b)
          end
    end
end;

{************************************************************************}
{  DATE: 7 Sep 84                                                        }
{  VERSION: 1.0                                                          }
{  NAME: adj_delay                                                       }
{  FUNCTION: This procedure adjust the simulation for the actual         }
{           network cable delay, if known                                }
{  INPUTS: none                                                          }
{  OUTPUTS: none                                                         }
{  GLOBAL VARIABLES: N                                                   }
{  GLOBAL TABLES USED: none                                              }
{  GLOBAL TABLES CHANGED: none                                           }
{  FILES READ: none                                                      }
{  FILES WRITTEN: none                                                   }
{  PROCEDURES CALLED: none                                               }
```

```
{  CALLING PROCEDURES: simulate                                          }
{  AUTHOR: Capt John M. Schriml                                          }
{  HISTORY: none                                                         }
{************************************************************************}


  procedure adj_delay;

    var delay,length,z:real;

    begin
      write ('Please enter, in nano seconds, the known around trip ');
      writeln ('delay of the bus.');
      readln (delay);
      writeln;
      write ('Please re-enter, in feet, the max round trip length ');
      writeln ('of the bus cable.');
      writeln ('<CR>');
      readln (length);
      z := (delay * 6564) /(10000.0 * length / 2);
      if z > 2
        then N := 1 + round(z)
        else N := 3;
      if N > 250
        then N := 250
    end;


{************************************************************************}
{  DATE: 30 Aug 84                                                       }
{  VERSION: 1.0                                                          }
{  NAME: display                                                         }
{  FUNCTION: Displays a copy of the data loaded when requested by the    }
{            user.                                                       }
{  INPUTS: none                                                          }
{  OUTPUTS: none                                                         }
{  GLOBAL VARIABLES: table, x                                            }
{  GLOBAL TABLES USED: table                                             }
{  GLOBAL TABLES CHANGED: none                                           }
{  FILES READ: none                                                      }
{  FILES WRITTEN: none                                                   }
{  PROCEDURES CALLED: none                                               }
{  CALLING PROCEDURES: simulate                                          }
{  AUTHOR: Capt John M. Schriml                                          }
{  HISTORY: none                                                         }
{************************************************************************}


  procedure display;

    begin
      writeln;
      write ('   TIME          BUS           NEXT       INPUT');
```

```
                    {* PROGRAM:EVALUATE *}

{************************************************************************}
{  DATE: 31 Aug 84                                                       }
{  VERSION: 1.0                                                          }
{  NAME: evaluate                                                        }
{  FUNCTION: Using the data generated by the simulation, this program    }
{            evaluates the results and displays them to the user.        }
{  INPUTS: none                                                          }
{  OUTPUTS: none                                                         }
{  GLOBAL VARIABLES: wild                                                }
{  GLOBAL TABLE USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                           }
{  FILES READ: none                                                      }
{  FILES WRITTEN: none                                                   }
{  PROCEDURES CALLED: get_data, show_parameters, sess  erformance,       }
{                     bus_operating_speed, display_data                  }
{  CALLING PROCEDURES: none                                              }
{  AUTHOR: Capt John M. Schriml                                          }
{  HISTORY: none                                                         }
{************************************************************************}


program evaluate (input,output);

  const propagation = 656400;   {ft/msec}

  type data = record
                sessions_id:integer;
                extended_id:integer;
                time_to_bus:integer4;
                interval_to_bus:integer4;
                time_before_next_input:integer4;
                input_interval:integer4;
                input_variance:integer4;
                busy_time_on_bus:integer4;
                traffic_delay:integer4;
                flow_control_delay:integer4;
                number_of_collisions:integer4;
                inputed_traffic:integer4;
                bus_busy_time:integer4;
                extra_time_to_tx:integer4;
                total_input_time:integer4
              end;

    bus_setup = record
                  bus_speed:integer4;
                  bus_length:integer4;
                  bus_overhead:integer
                end;

    clock_time = record
```

total-bus-busy-time (G)(I4) - Represents the total amount of time the bus was bus for the entire network. Time is in T intervals.

total-collisions (G)(I4) - Represents the total amount of collisions for the entire network.

wild (G)(C) - Represents user's response to a question.

wild-card (L)(C) - Represents user's response to a question.

x (G)(I) - Is used as a counter in loops.

z (L)(I) - Is used as a counter in loops.

The actual Pascal program, Evaluate, begins on the next page.

temp1 (G)(I4) - Represents in usec the amount of delay due to traffic on the bus, for a given session.

temp2 (G)(I4) - Represents in usec the amount of delay due to collisions, for a given session.

temp3 (G)(I4) - Represents in usec the amount of delay in due to flow control, for a given session.

temp6 (G)(R) - Represents the total amount of delay, for a given type of session, due to the bus being busy. Delay is in T intervals.

temp7 (G)(R) - Represents the total amount of delay, for a given type of session, due to flow control. Delay is in T intervals.

temp9 (G)(I4) - Represents the total amount of inputs for a given type of session.

temp11 (G)(R) - Represents the total amount of delay, for a given type of session, due to collisions. Delay is in T intervals.

temp12 (G)(I) - Represents the ID number of a session type.

temp14 (G)(I4) - Represents in bits/sec the data rate of a session's terminal.

temp15 (G)(I4) - Represents the number characters inputed by each input.

temp16 (G)(I4) - Represents in msec the time interval between inputs.

temp17 (G)(I4) - Represents the number of times that a given session was repeated in the simulation.

temp18 (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

time (G)(I4) - Represents the transition time or simulation time in T intervals.

table[x,7] (G)(I4) - Represents the summation of the amount of waiting time for inputs because the BIU was having trouble passing the traffic it had. Time is in T intervals.

table[x,8] (G)(I4) - Represents the summation of the number of collisions a session had.

table[x,9] (G)(I4) - Represents the summation of the number of inputs made by a session.

table[x,10] (G)(I4) - Represents the summation of the time that the bus was busy because of a particular session. Time is in T intervals.

table[x,11] (G)(I4) - Represents the summation of the amount of extra waiting time because of collisions. Time is in T intervals.

table[x,12] (G)(I4) - Represents the actual time between inputs. Time is in T intervals.

tem (L)(I4) - Represents the number of inputs made by each session.

tem2 (G)(I4) - Represents in usec the actual time interval between inputs for each type of session. This time interval was determined from the results of monitoring the amount of traffic being allowed in by flow control.

tem3 (G)(I4) - Represents in usec the actual time interval between inputs for each type of session. This time interval was determined from the results of monitoring the amount of traffic being passed by the terminals.

tem6 (L)(R) - Represents to total amount of waiting time for one type of session, due to the bus being busy. Time is in T intervals.

tem7 (L)(R) - Represents the total amount of waiting time due to flow control and waiting time between inputs, for a particular type of session.

sim.number-of-collisions (G)(I4) - Represents the number of collisions of each session.

sim.sessions-id (G)(I) - Represents the ID number of a session.

sim.time-before-next-input (G)(I4) - Represents the amount of time that must elapse before the next input is made. Time is in T intervals.

sim.time-to-bus (G)(I4) - Represents the amount of time that must elapse before an input reaches the BIU. Time is in T intervals.

sim.total-input-time (G)(I4) - Represents the actual amount of time between inputs. Time is in T intervals.

sim.traffic-delay (G)(I4) - Represents the amount of time traffic is delayed because the bus is busy. Time is in T Intervals.

T - This is not an actual variable used in this program. T represents the time interval of the simulation clock. T is equal to the amount time that equates to 1/2 round trip propagation delay of the bus cable.

table[x,1] (G)(I4) - Represents the amount of time that must elapse before the input reaches the BIU. Time is in T intervals.

table[x,2] (G)(I4) - Represents the calculated time it takes an input to travel from the terminal to BIU. Time is in T intervals.

table[x,3] (G)(I4) - Represents the amount of time that must elapse before the next input is made. Time is in T intervals.

table[x,4] (G)(I4) - Represents the calculated time between inputs. Time is in T intervals.

table[x,5] (G)(I4) - Represents the calculated time that the bus needs to pass a session's input. Time is in T intervals.

table[x,6] (G)(I4) - Represents the summation of the amount of waiting time due to the bus being busy. Time is in T intervals.

rate (G)(I4) - Represents in bits/sec the data rate of the bus.

real-time (G)(I4) - Represents the real time of the transition and simulation times. Real-time is in msec.

sim (G) - Represents a record of a sessions parameters and performance data.

sim.bus-busy-time (G)(I4) - Represents the amount of time for each session that the bus was busy. Time is in T intervals.

sim.busy-time-on-bus (G)(I4) - Represents the calculated amount time the bus is busy for each input. Time is in T intervals.

sim.extended-id (G)(I) - Represents the extended ID of a session.

sim.extra-time-to-tx (G)(I4) - Represents the additional amount of waiting time traffic must wait because of collisions. Time is in T intervals.

simfile (G) - Represents the file containing the records of each session's parameters and performance data.

sim.flow-control-delay (G)(I4) - Represents the amount of time inputs were delayed because flow control was employed. Time is in T intervals.

sim.input-interval (G)(I4) - Represents the calculated time interval between inputs. Time is in T intervals.

sim.inputed-traffic (G)(I4) - Represents the number of inputs made by a session.

sim.input-variance (G)(I4) - Represents the maximum +/- deviation of the time interval between inputs. Time is in T intervals.

sim.interval-to-bus (G)(I4) - Represents the calculated amount of time it takes an input to travel from the terminal to BIU. Time is in T intervals.

each input.

net.quantity (G)(I4) - Represents the number of times a session is duplicated in the simulation run.

net.speed (G)(I4) - Represents in bits/sec the data rate of a session's terminal.

net.variance (G)(I4) - Represents in msec the maximum +/- deviation of the interval between inputs.

net-collisions (G)(I4) - Represents the average number of collisions per second.

net-input-speed (G)(I4) - Represents in bits/sec the amount of traffic the users are attempting to input.

net-flow-control- rate (G)(I4) - Represents in bits/sec the amount of traffic that the user is actually inputing. This traffic is determined from the results of monitoring the amount of time that flow control is employed.

net-traffic rate (G)(I4) - Represents in bits/sec the amount of traffic that the user is actually inputing. This traffic is determined from the results of monitoring the amount traffic being passed by the terminals.

net-terminal-speed (G)(I4) - Represents in bits/sec the maximum amount of traffic that the terminals are physically able to pass.

overhead (G)(I) - Represents the amount of management bits added by the BIU to the information bits of the user.

range (G)(I) - Represents the number of sessions simulated. Range is determined from counting the number of sessions loaded into the program. Range is used in loops and to limit the size of the matrix table.

clock.delay (G)(I4) - Represents the actual bus cable delay used by the simulation run. Delay is in T intervals.

clockfile (G) - Represents the file containing the clock record.

clock.percentage (G)(I) - Represents the persistant percentage, if p-persistant was simulated.

clock.persistant (G)(I) - Represents the type of persistant used for the simulation.

clock.sclck (G)(I4) - Represents the amount of simulation time used for collecting data. Time is in T intervals.

clock.tclck (G)(I4) - Represents the amount of simulation time used for transistion before data was collected. Time is in T intervals.

ext-id[x] (G)(I) - Represents a session's extended ID for a given x.

idle (L)(R) - Represents the amount of time a terminal is idle during a session. Time is in T intervals.

id[x] (G)(I) - Represents the session's ID number for a given x.

k (L)(R) - Is used for intermediate calculations.

l (L)(R) - Is used for intermediate calculations.

length (G)(I4) - Represents in feet the one way length of the bus cable.

net (G) - Represents a record of a network session.

netfile (G) - Represents a file of records containing network sessions.

net.id (G)(I) - Represents the ID number of a session.

net.interval (G)(I4) - Represents in msec the time interval between inputs.

net.number (G)(I4) - Represents the number of characters inputed by

PART B7: Pascal Program Evaluate

The variables and their purpose, as used with the program Evaluate, are as follows:

VARIABLES:

bus (G) - Represents a record containing the bus information inputed by the user.

bus.bus-length (G)(I4) - Represents in feet the one way length of the bus cable.

bus.bus-overhead (G)(I4) - Represents the amount of management bits (overhead) added by the BIU to the user's information bits.

bus.bus-speed (G)(I4) - Represents in bits/sec the data rate speed of the bus.

busfile (G) - Represents the file containing the record of the bus infomation inputed by the user.

bus-flow-control-rate (G)(I4) - Represents in bits/sec the amount of traffic being passed on the bus by using the results of monitoring the amount of time flow control is employed.

bus-rate (G)(I4) - Represents in bits/sec the amount of traffic being passed on the bus by using the results of monitoring the bus' busy time.

bus-traffic-rate (G)(I4) - Represents in bits/sec the amount of traffic being passed on the bus by using the results of monitoring the amount of traffic the terminals were passing.

clock (G) - Represents a record of the clock, delay, percentage, and persistant information, provided by the user.

```
    rewrite (clockfile);
    clc.T_click := tclock;
    clc.S_click := sclock;
    clc.delay := N;
    clc.persistant := persistant;
    clc.percentage := percentage;
    clockfile^ := clc;
    put (clockfile);
    close (clockfile)
end.
```

```
      then collision
      else if (event_flag ='b') and (waiting >1) and
              (persistant = 2)
        then
          begin
            adj_further;
            if waiting > 1
              then collision
          end;
  if event_flag = 'i'
    then adj_input
until clock < 0;
tclock := tclock - clock;
clock := sclock;
for x := 1 to range do
  begin
    table[x,6] := 0;
    table[x,7] := 0;
    table[x,8] := 0;
    table[x,9] := 0;
    table[x,10] := 0;
    table[x,11] := 0;
    table[x,15] := 0
  end;
repeat
  writeln (clock);
  find_next_event;
  clock := clock - (next_event - 1);
  if (screen = 'y') or (screen ='Y')
    then test;
  if next_event <> 1
    then prepare_for_event;
  if (screen = 'y') or (screen = 'Y')
    then test;
  if (event_flag = 'b') and (waiting = 1)
    then adj_bus
    else if (event_flag = 'b') and (waiting > 1) and
                  (persistant = 1)
      then collision
      else if (event_flag = 'b') and (waiting > 1) and
                  (persistant = 2)
        then
          begin
            adj_further;
            if waiting > 1
              then collision
          end;
  if event_flag = 'i'
    then adj_input
until clock <0;
sclock := sclock - clock;
writeln ('Actual simulation clock time is ', sclock);
file_results;
```

```
writeln (range, ' sessions were loaded for this simulation run.');
writeln;
writeln ('Do you wish to review the data loaded? (Y/N)');
readln (answer);
writeln;
if (answer = 'y') or (answer = 'Y')
  then display;
writeln;
write ('Is the actual round trip delay of the network known? ');
writeln ('(Y/N)');
writeln ('<CR>');
readln (answer);
if (answer = 'y') or (answer = 'Y')
  then adj_delay;
limitset := [1..N];
writeln;
repeat
  writeln ('Is this simulation a 1- or p- persistant?');
  writeln ('If 1-persistant enter 1, if p-persistant enter 2.');
  writeln('<CR>');
  readln (persistant);
until (persistant = 2) or (persistant = 1);
if persistant = 2 then
  begin
    writeln ('Enter percentage bewteen 1  and 100.');
    readln (percentage)
  end;
writeln;
writeln ('Please enter the transition clock time.');
readln (tclock);
writeln ('Please enter the simulation clock time.');
readln (sclock);
writeln ('Please enter the time of day.');
writeln ('(example: If time is 13:15, enter 1315)');
readln (seed);
clock := tclock;
writeln ('Do you need a display of the simulation run? (Y/N)');
writeln ('<CR>');
readln (screen);
repeat
  writeln (clock);
  find_next_event;
  clock := clock - (next_event - 1);
  if (screen = 'y') or (screen = 'Y')
    then test;
  if (next_event <> 1)
    then prepare_for_event;
  if (screen = 'y') or (screen = 'Y')
    then test;
  if (event_flag = 'b') and (waiting = 1)
    then adj_bus
    else if (event_flag = 'b') and (waiting > 1) and
              (persistant = 1)
```

```
          sim.inputed_traffic := table[x,9];
          sim.bus_busy_time := table[x,10];
          sim.extra_time_to_tx := table[x,11];
          sim.total_input_time := table[x,15];
          simfile^ := sim;
          put (simfile)
        end;
        close (datafile);
        close (simfile)
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: test                                                             }
{  FUNCTION: To display to the user each step of the simulation upon      }
{            request.  Normal used for trouble shooting.                  }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: x, table, range                                      }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: simulate                                           }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure test;

    begin
      writeln;
      for x := 1 to range do
        begin
          write (table[x,1]:6,table[x,2]:6,table[x,3]:7,table[x,4]:7);
          write (table[x,5]:5,table[x,6]:7,table[x,7]:7,table[x,8]:3);
          write (table[x,9]:3,table[x,10]:7,table[x,11]:5);
          writeln (table[x,12]:3,table[x,13]:5,table[x,14]:5)
        end
    end;

  begin
    assign (clockfile,'clock');
    assign (datafile,'net_data');
    assign (simfile,'simulate');
    N := 3;
    percentage := 100;
    get_data;
    writeln;
```

```
      writeln ('        INPUT      BUS');
      write ('  TO BUS       INTERVAL       INPUT     INTERVAL');
      writeln ('      VARIANCE   BUSY TIME');
      writeln;
      for x := 1 to range do
        begin
          write (table[x,1]:9,table[x,2]:13,table[x,3]:12);
          writeln (table[x,4]:11,table[x,14]:12,table[x,5]:10);
        end
    end;


{*************************************************************************}
{  DATE: 30 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: file_results                                                     }
{  FUNCTION: Stores results of the simulation run.                        }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: simfile, datafile, x, comp, sim, table               }
{  GLOBAL TABLES USED: table                                              }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: net_data                                                   }
{  FILES WRITTEN: simulate                                                }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: simulate                                           }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure file_results;

    begin
      rewrite (simfile);
      reset (datafile);
      x := 0;
      while not eof (datafile) do
        begin
          comp := datafile^;
          get (datafile);
          x := x + 1;
          sim.session_id := comp.session_id;
          sim.extended_id := comp.extended_id;
          sim.time_to_bus := table[x,1];
          sim.interval_to_bus := table[x,2];
          sim.time_before_next_input := table[x,3];
          sim.input_interval := table[x,4];
          sim.input_variance := table[x,14];
          sim.busy_time_on_bus := table[x,5];
          sim.traffic_delay := table[x,6];
          sim.flow_control_delay := table[x,7];
          sim.number_of_collisions := table[x,8];
```

```
                    tclck,clck:integer4;
                    delay,persistant,percentage:integer
                  end;

    net_sessions = record
                     id:integer;
                     speed,number,interval,variance,quantity:integer4
                   end;

  var id:array[1..,500] of integer;
      ext_id:array[1..500] of integer;
      table:array[1..500,1..12] of integer4;
      x,range,overhead,temp12:integer;
      temp14,temp15,temp16,temp17,temp18,rate,time,length,real_time,
      net_terminal_speed,net_input_speed,tem2,temp1,temp2,temp3,
      net_traffic_rate,net_flow_control_rate,bus_traffic_rate,
      net_delay_due_to_collisions,bus_flow_control_rate,
      total_collisions,net_collisions,total_bus_busy_time,
      bus_rate,temp9,tem3:integer4;
      temp6,temp7,temp11:real;
      wild:char;
      sim:data;
      bus:bus_setup;
      net:net_sessions;
      clock:clock_time;
      simfile:file of data;
      busfile:file of bus_setup;
      clockfile:file of clock_time;
      netfile:file of net_sessions;


{*************************************************************************}
{  DATE: 31 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: get_data                                                         }
{  FUNCTION: To load the results of simulation into a matrix for          }
{            evaluation.                                                  }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: x, simfile, id, ext_id, table, sim, range            }
{  GLOBAL TABLES USED: id, ext_id, table                                  }
{  GLOBAL TABLES CHANGED: id, ext_id, table                               }
{  FILES READ: simulate                                                   }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: evaluate                                           }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure get_data;
```

```
    begin
      reset (simfile);
      x := 0;
      while not eof (simfile) do
        begin
          sim := simfile^;
          get (simfile);
          x := x + 1;
          id[x] := sim.sessions_id;
          ext_id[x] := sim.extended_id;
          table[x,1] := sim.time_to_bus;
          table[x,2] := sim.interval_to_bus;
          table[x,3] := sim.time_before_next_input;
          table[x,4] := sim.input_interval;
          table[x,5] := sim.busy_time_on_bus;
          table[x,6] := sim.traffic_delay;
          table[x,7] := sim.flow_control_delay;
          table[x,8] := sim.number_of_collisions;
          table[x,9] := sim.inputed_traffic;
          table[x,10] := sim.bus_busy_time;
          table[x,11] := sim.extra_time_to_tx;
          table[x,12] := sim.total_input_time
        end;
      range := x;
      close (simfile)
    end;


{*************************************************************************}
{  DATE: 31 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: show_parameters                                                  }
{  FUNCTION: To show the parameters for which the results of the          }
{            simulation are based on.                                     }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: busfile, bus, rate, length, overhead, clock,         }
{                    clockfile, time, real_time                           }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: clock, bus                                                 }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: evaluate                                           }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure show_parameters;

    var k,delay:real;
```

```
begin
  write ('The results of the simulation run are based on the ');
  writeln ('following network parameters.');
  writeln;
  reset (busfile);
  bus := busfile^;
  get (busfile);
  write ('Data rate of the bus was ',bus.bus_speed:4);
  writeln (' bits/sec.');
  writeln;
  length := bus.bus_length;
  write ('Max round trip length of the bus was ',(length * 2):4);
  writeln (' feet.');
  reset (clockfile);
  clock := clockfile^;
  get (clockfile);
  writeln;
  if clock.persistant = 1
    then writeln ('1-persistant was used.')
    else writeln ('P-persistant was used with a percentage ',
                  'of ',clock.percentage:2,' %.');
  delay := (((clock.delay - 1) * float4(length)) * 1000000) /
           propagation;
  writeln;
  write (' The round trip delay of the bus cable was ');
  writeln (trunc4(delay):3,' nano seconds.');
  writeln;
  writeln ('The BIU overhead was ',bus.bus_overhead:3,' bits.');
  writeln;
  rate := bus.bus_speed;
  overhead := bus.bus_overhead;
  close (busfile);
  writeln ('The actual transition clock time was ',
                                    clock.tclck:6,'.');
  writeln;
  writeln ('The actual data collecting clock time was ',
                                    clock.clck:6,'.');
  writeln;
  time := clock.tclck;
  k := (float4(time) * length)/propagation;
  real_time := trunc4(k);
  write ('The transition clock time approx equals ',real_time:3);
  writeln (' milliseconds(msec) of real ');
  writeln ('network run time.');
  time := clock.clck;
  close (clockfile);
  k := (float4(time) * length)/propagation;
  real_time := trunc4(k);
  writeln;
  write ('The data collecting clock time approx equals ');
  writeln (real_time:3,' milliseconds(msec) of ');
  writeln ('real network run time.');
  writeln
```

```
    end;

{*************************************************************************}
{  DATE: 31 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: bus_operating_speed                                              }
{  FUNCTION: To display to the user the amount of traffic the bus is      }
{            passing and how much traffic is being inputed to the         }
{            network.                                                     }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: rate, bus_rate, bus_traafic_rate, net_collisions     }
{                    bus_flow_control_rate, net_terminal_speed,           }
{                    net_traffic_rate, net_input_speed,                   }
{                    net_flow_control_rate                                }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: none                                               }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure bus_operating_speed;

    begin
      writeln;
      writeln ('       *** BUS PERFORMANCE ***');
      writeln;
      writeln ('The bus is set up to operate at ',rate:6,
                                          ' bits/sec.');
      writeln ('Traffic on the bus indicates a rate of ',bus_rate:6,
                                          ' bits/sec.');
      writeln ('Traffic being passed by the terminals indicate a ',
                        'rate of ',bus_traffic_rate:6,' bits/sec.');
      writeln;
      writeln ('Traffic being inputed by the user indicate a bus rate',
                      ' of ',bus_flow_control_rate:6,' bits/sec.');
      writeln ('Collisions were at the rate of ',net_collisions:1,
                    ' collisions per sec.');
      writeln;
      writeln ('        *** NETWORK INPUT TRAFFIC ***');
      writeln;
      writeln ('The MAX traffic rate which the terminals can pass is ',
                  net_terminal_speed:3,' bits/sec.');
      writeln ('The simulation indicates that the terminals were ',
                'passing traffic at ',net_traffic_rate:3,' b/s.');
      writeln ('The users are attempting to input ',net_input_speed:3,
                           ' bits/sec.');
```

```
      writeln ('The simulation indicates that the network is allowing',
                  ' an input of ',net_flow_control_rate:4,' b/s.');
    end;


{*************************************************************************}
{  DATE: 31 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: show_performance                                                 }
{  FUNCTION: To display the peroformance of each individual type of       }
{            sessions to the user.                                        }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: temp17, temp12, temp14, temp15, temp16, temp18,      }
{                    temp1, temp3, temp2, tem3, tem2, tem1                }
{  GLOBAL TABLES USED: none                                               }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: sessions_performance                               }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure show_performance;

    var l,k:real;

    begin
      write (temp17:1,' typical sessions were set up as ');
      writeln ('follows:');
      writeln ('ID: ',temp12:1);
      writeln ('TERMINAL RATE: ',temp14:3,' bits/sec');
      writeln ('# of CHARACTERS PER INPUT: ',temp15:1);
      writeln ('TIME INTERVAL BETWEEN INPUTS: ',temp16:2,' msec',
                  '      var ', temp18:1,' msec');
      writeln ('*AVE MAX INPUT RATE: ',(8 * temp15 * 1000) div temp16,
                                  ' bits/sec*');
      writeln;
      writeln ('The performance was as follows.');
      writeln;
      writeln ('DELAY PER INPUT:');
      writeln ('    -due to traffic on the bus: ',temp1:3,' usec');
      writeln ('    -due to collisions: ',temp3:3,' usec');
      writeln ('    -due to flow control: ',temp2:3,' usec');
      writeln;
      writeln ('THOUGHPUT RATE PER SESSION:');
      writeln ('   -time interval between inputs:');
      writeln ('      --terminals indicate ',tem3:2,' usec');
      writeln ('      --inputs indicate ',tem2:2,' usec');
      k := (1000000.0 * 8 * temp15)/tem3;
```

```
      l := (1000000.0 * 8 * temp15)/tem2;
      writeln ('   -transmission rate:');
      writeln ('      --terminals indicate ',trunc4(k):2,' bits/sec');
      writeln ('      --inputs indicate ',trunc4(l):2,' bits/sec')
    end;


{*************************************************************************}
{  DATE: 31 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: sessions_performance                                             }
{  FUNCTION: To do the actual performance calculations.                   }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: net_traffic_rate, net_flow_control_rate,             }
{                    bus_traffic_rate, bus_flow_control_rate,             }
{                    net_collisions,                                      }
{                    net_input_speed, net_terminal_speed, netfile,        }
{                    total_collisions, total_bus_busy_time, net,          }
{                    temp6,temp7, temp9, temp11, id, table, temp1,        }
{                    tem1, tem3, temp3, tem2, temp2, temp12, temp14,      }
{                    temp15, temp16, temp17, temp18, bus_rate             }
{  GLOBAL TABLES USED: id, table                                          }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: network                                                    }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: show_performance                                    }
{  CALLING PROCEDURES: evaluate                                           }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure sessions_performance;

    var wild_card:char;
        k:real;
        z:integer;
        tem:integer4;
        idle,tem6,tem7:real;

    begin
      net_traffic_rate := 0;
      net_flow_control_rate := 0;
      bus_traffic_rate :=0;
      bus_flow_control_rate := 0;
      net_collisions := 0;
      net_input_speed := 0;
      net_terminal_speed := 0;
      total_collisions := 0;
      total_bus_busy_time := 0;
      write ('The performance of various types of sessions are');
      writeln (' as follows:');
```

```
writeln;
writeln ('Please enter <CR> to continue.');
read (wild_card);
writeln;
reset (netfile);
while not eof (netfile) do
  begin
    net := netfile^;
    get (netfile);
    temp6 := 0;
    tem6 := 0;
    temp7 := 0;
    tem7 := 0;
    temp9 := 0;
    temp11 := 0;
    for z := 1 to range do
      if id[z] = net.id then
        begin
          tem := table[z,9];
          temp6 := temp6 + table[z,6];
          tem6 := tem6 + (table[z,2] * tem) + table[z,6] +
                    table[z,11];
          temp7 := temp7 + table[z,7];
          tem7 := tem7 + table[z,12] + table[z,7];
          total_collisions := total_collisions + table[z,8];
          total_bus_busy_time := total_bus_busy_time +
                                 table[z,10];
          temp9 := temp9 + table[z,9];
          temp11 := temp11 + table[z,11]
        end;
    k := (temp6 * float4(length) * 1000)/(float4(temp9) *
                          propagation);
    temp1 := trunc4(k);
    idle := ((time * float(net.quantity)) - tem6)/temp9;
    k := ((idle + (tem6 / temp9)) * length * 1000)/propagation;
    tem3 := trunc4(k);
    k := (temp7 * float4(length) * 1000)/(float4(temp9) *
                          propagation);
    temp2 := trunc4(k);
    k := (tem7 * float4(length) * 1000)/(float4(temp9) *
                          propagation);
    tem2 := trunc4(k);
    k := (temp11 * float4(length) * 1000)/(float4(temp9) *
                          propagation);
    temp3 := trunc4(k);
    k := (1000000 * float4(net.number) * net.quantity * 8)/tem3;
    net_traffic_rate := net_traffic_rate + trunc4(k);
    k := (1000000 * float4(net.number) * net.quantity * 8)/tem2;
    net_flow_control_rate := net_flow_control_rate + trunc4(k);
    k := (((net.number * 8) + overhead) * float4(net.quantity) *
                         1000000.0)/tem3;
    bus_traffic_rate := bus_traffic_rate + trunc4(k);
    k := (((net.number * 8) + overhead) * float4(net.quantity) *
```

```
                                1000000.0)/tem2;
            bus_flow_control_rate := bus_flow_control_rate + trunc4(k);
            temp12 := net.id;
            temp14 := net.speed;
            temp15 := net.number;
            temp16 := net.interval;
            temp18 := net.variance;
            temp17 := net.quantity;
            net_terminal_speed := net_terminal_speed + (temp14 * temp17);
            net_input_speed := net_input_speed + (((temp15 * 1000) * 8 *
                                temp17) div temp16);
            show_performance;
            writeln;
            writeln ('Please enter<CR> to continue');
            read (wild_card);
            writeln;
          end;
        k := ((total_bus_busy_time * 1.0) * rate)/time;
        bus_rate := trunc4(k);
        net_collisions := (total_collisions * 1000) div real_time
      end;


{*************************************************************************}
{  DATE: 31 Aug 84                                                        }
{  VERSION: 1.0                                                           }
{  NAME: display_data                                                     }
{  FUNCTION: Displays raw data upon user request.                          }
{  INPUTS: none                                                           }
{  OUTPUTS: none                                                          }
{  GLOBAL VARIABLES: id, ext_id, table                                    }
{  GLOBAL TABLES USED: id, ext_id, table                                  }
{  GLOBAL TABLES CHANGED: none                                            }
{  FILES READ: none                                                       }
{  FILES WRITTEN: none                                                    }
{  PROCEDURES CALLED: none                                                }
{  CALLING PROCEDURES: evaluate                                           }
{  AUTHOR: Capt John M. Schriml                                           }
{  HISTORY: none                                                          }
{*************************************************************************}


  procedure display_data;

    var z:integer;

    begin
      writeln;
      writeln ('        *** RAW DATA ***');
      writeln ;
      write ('                  WAITING    FLOW      ');
      writeln ('          BUS    EXTRA TIME DUE');
      write (' ID   EXT ID     FOR BUS   CONTROL    ');
```

```
    writeln (' INPUTS   BUSY   TO COLLISIONS');
    for z := 1 to range do
      begin
        write (id[z]:3,ext_id[z]:7,table[z,6]:14,table[z,7]:10);
        write (table[z,9]:9,table[z,10]:9);
        writeln (table[z,11]:13)
      end;
    writeln;
    writeln ('There were ', total_collisions:2,' collisions.')
  end;

begin
  assign (simfile,'simulate');
  assign (busfile,'bus');
  assign (clockfile,'clock');
  assign (netfile,'network');
  get_data;
  show_parameters;
  sessions_performance;
  bus_operating_speed;
  writeln;
  writeln ('Please enter <CR> to continue');
  read (wild);
  writelr;
  writeln ('Do you wish to see a copy of the raw data? (Y/N)');
  writeln ('<CR>');
  read(wild);
  if (wild = 'y') or (wild = 'Y')
    then display_data
end.
```

APPENDIX C: SIMULATION MODEL'S INPUT LIMITATIONS

The following input limitations are for the simulation model developed by this thesis, using a 16 bit minicomputer. Some limitations may change if the computer programs are loaded into a computer with more than 128K of memory.

Simulator's Basic Unit of Time. Since the simulation model takes into account numerous variables, many of the limitations can not be determined until some of the variables are selected. One of the major factors contributing to the model's limitations is the fact that the basic unit of time for the simulation is 1/2 the propagation delay of the cable. For this particular simulation model, this implies that all inputs transmitted on the bus, must take at least the time equivalent to 1/2 the propagation delay of the cable for the simulation to be successful. The basic unit of time, for the simulator, leads to the following equation:

$$\frac{((\#\ \text{char/input in bits}) + (\text{overhead in bits})) * 656400000\ \text{ft/sec}}{(\text{bus rate in bits/sec}) * (1/2\ \text{cable length in ft})} \geq 1$$

The number 1 in the equation represents one tick of the simulation clock. As long as the equation is equal to or greater than 1, for a given bus rate, cable length, # characters/input, and overhead bits, the simulation model should perform correctly.

Number of Sessions. As mentioned previously, the memory capability of the micro-computer limits the number of sessions to 500.

Terminal Speed. The only physical limitation on the value of the terminal speed is the value of an integer[4]. The model was designed with the assumption that the bus is able to pass traffic much faster than the terminal. The model will not perform correctly if the terminal speed is faster than the bus speed.

Simulation Time. The maximum simulation time, with respect to the ticks of the simulation clock, is 2,147,483,648. In the case of the AFLC network, this equates to approximately 55 minutes of real simulation time. The maximum simulation time was used one time, with one session, and it required 25 minutes of computer time to simulate the full 55 minutes of real simulation time. The actual required computer time will vary with the number of sessions being simulated.

Time Interval Between Inputs. The minimum allowable interval between inputs is 1 msec. The maximum interval between inputs is 3,271,000 msec, with respect to the AFLC network. The maximum value is a little unrealistic and would allow for only one input.

Actual Cable Delay. The simulation model will accept values for known network delay between 1 and 125 times the propagation delay of the cable. The user entering known network delays outside this range, will cause the simulation model to default to the appropriate extreme end of the range.

Bibliography

1. O'Reilly, P. J. P. and J. L. Hammond. "An Efficient Alogorithm for Generating the Busy/Idle Periods of A Channel Using CSMA and Loaded by an Arbitrary Number of Stations," Proc COMPCON , 427-436, Washington DC , Sep 82.

2. Tokoro, M. and K. Tamaru. "Acknowledge Ethernet," COMPCON , 320-325, Fall 77.

3. Hughes, H. D. and L. Li. "A Simulation Model of the Ethernet," Technical Report TR #82-008 , Dept of Computer Science, Michigan State University, 82.

4. Almes, G. T. and E. D. Lazowski. "The Behavior of Ethernet-Like Computers Communications Networks", Proc &th Symposium on OS Principles , 66-81, Dec 82.

5. Tobagi, F. A. and V. B. Hunt, "Perfomance Analysis of Carrier Sense Multiple Access with Collision Detection", Computer Networks , 4 :245-259, (80).

6. FitzGerald, Jerry. Business Data Communications. New York: John Wiley & Sons, 1984.

7. Tanenbaum, Andrew S. Computer Networks. Englewood Cliffs: Prentice-Hall, 1981.

8. Hopkins, G. T. "Multimode Communications on the Mitrenet," Computer Networks , 4 : 229-233 (1980).

9. "Special Report: Local Network Review and Projection," The LocalNetter Newsletter , 1 (1): 55.3-55.7 (Jan 83).

10. Byers, T. J. "Electronic Ties That Bind," Computers and Electronics , 22 (3): 68-73 (Mar 84).

11. Metcalfe, Robert M. "Ethernet: Distribution Packet Switching for Local Computer Networks", Communications of the ACM , 19 (7): 395-404 (Jul 76).

12. Schruben, Lee "Modeling Systems Using Discrete Event Simulation", Proc Simulation Conference , 101-107, Winter 83.

13. Seila, Andrew F. and Der-Fa Robert Chen, "Discrete Event Simulation on Mini- and Microcomputers: Some Experiments with the Pascal Language", Proc Simulation Conference , 41-43, Winter 81.

14. Bux, Werner. "Local Area Subnetworks: A Performance Comparison", IEEE Transcations on Communications , 29 (10):1465-1473 (Oct 81).

VITA

Captain John M. Schriml was born 2 June 1949 in Dayton, Ohio. He served as a Nike Hercules Missile Repairman with the Army for three years, after graduating from Chaminade High School in Dayton. Upon his separation from the Army, Captain Schriml returned to Dayton and attended the University of Dayton from which he received the degree of Bachelor of Electrical Engineering in December 1974. He received his commission in the USAF through the Air Force's OTS program. Captain Schriml's first assignment was as a HF (High Frequency) Radio Test and Evaluation Team Chief in the Pacific. His next assignment was with HQ AFCC as the AUTOVON Test and Evaluation Team Chief, which required numerous worldwide TDYs. Captain Schriml's last assignment prior to attending the School of Engineering, AFIT, was with the 1815 Test and Evaluation Squadron where he was the Section Chief for AFCC's Operational Test and Evaluation Section.

Permanent Address: 1845 Pershing Blvd

Dayton, Ohio 45420

REPORT DOCUMENTATION PAGE

CURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
ICATION/DOWNGRADING SCHEDULE	
IG ORGANIZATION REPORT NUMBER(S) IT/GE/ENG/84D-57	5. MONITORING ORGANIZATION REPORT NUMBER(S)
ERFORMING ORGANIZATION hool of Engineering	6b. OFFICE SYMBOL (If applicable) AFIT/ENG
7a. NAME OF MONITORING ORGANIZATION	
City, State and ZIP Code) r Force Institute of Technology ight-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
UNDING/SPONSORING ATION AFLC	8b. OFFICE SYMBOL (If applicable) SYC
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
City, State and ZIP Code) ight-Patterson AFB, Ohio 45433	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
lude Security Classification) e Box 19	
AUTHOR(S) hn M. Schriml, B.E.E, Capt, USAF	
REPORT Thesis	13b. TIME COVERED FROM _____ TO _____
14. DATE OF REPORT (Yr., Mo., Day) 1984 December	15. PAGE COUNT 180
NTARY NOTATION	

COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
GROUP	SUB. GR	Communications Network, Simulation, CSMA/CD, Bus, Local Networks CATV
02		

(Continue on reverse if necessary and identify by block number)

tle: SIMULATION MODEL OF A CSMA/CD BUS LOCAL AREA NETWORK WITH MULTIPLE VARIABLES

esis Advisor: Walter D. Seward, Major, USAF

Approved for public release: IAW AFR 190-17.
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

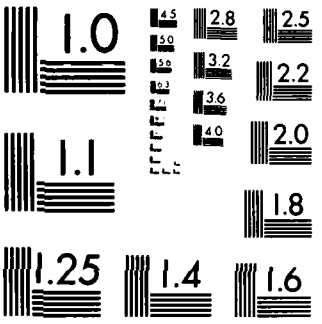
TION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
D/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED	
RESPONSIBLE INDIVIDUAL alter D. Seward, Major, USAF	22b TELEPHONE NUMBER (Include Area Code) 513-255-5533	22c OFFICE SYMBOL AFIT/ENG

473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A151 706 SIMULATION MODEL OF A CSMA/CD BUS LOCAL AREA NETWORK WITH MULTIPLE VARIABLES(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J M SCHRIML 3/3

UNCLASSIFIED DEC 84 AFIT/GE/ENG/84D-57 F/G 12/1 NL



Numerous statistical studies of the expected performance of a CSMA/CD bus local area network have been completed. With the statistical approach, the number of input sources is normally limited. Also, input sources normally possess the same statistical parameters with respect to input rate and the amount of data per input. The CSMA/CD bus local area network simulation model developed by this thesis can handle up to 500 input sources, all with a different input rate and amount of data per input. The limitation of 500 is due only to the fact that the simulation model was implemented on a 128K memory micro-computer. The number of sources can be increased by implementing the simulation model with a computer with a larger memory.

The simulation model takes the approach that once an input is made, the time for the input to travel through the various stages of a network can be easily calculated. Therefore, the simulation model generates traffic based on the statistical parameters of each individual source, then tracks the input as the simulation clock ticks. Using the memory power of the computer to keep track of the location of all inputs, the simulation model is able to determine the effect of an input on all other inputs. In some cases, an input has no direct effect on other inputs, and at the other extreme, when inputs want to use the bus at the same time, they have a drastic effect on each others performance. Numerous tests were performed to demonstrate the ability of the simulation model to model a CSMA/CD bus LAN. The simulation model will accept the following multiple variables prior to each simulation run: data rate of the bus, length of the bus cable, overhead bits of the bus, actual delay of the bus cable, data rate of each source terminal, time interval between each input, amount of data per each input, and whether the CSMA/CD is 1-persistant or p-persistant.

END

FILMED

5-85

DTIC